Programación entera, el método del árbol de cubos, su algoritmo paralelo y sus aplicaciones

Dr. José Crispín Zavala Díaz[1], Dr. Vladimir Khachaturov[2]
[1] Facultad de Contabilidad, Administración e Informática, jc_zavala2002@yahoo.com
[2] Facultad de Ciencias, vr_khach@yahoo.com
Universidad Autónoma del Estado de Morelos

1. Introducción.

Frecuentemente los problemas de optimización discreta se presentan de muy diversa índole en la administración. Tal es el caso del área de la administración de las cadenas de suministro, donde la distribución de materiales y objetos tienen una fuerte influencia en las finanzas, la contabilidad, los sistemas de información, *marketing* y compras. Además de lo anterior, esta distribución de materiales y objetos pueden constituir una ventaja competitiva para una empresa [1].

Algunos sostienen que, en condiciones ideales, una persona dentro de la empresa debe de tomar todas las decisiones sobre la administración de los materiales, porque las áreas mencionadas en el párrafo superior están estrechamente relacionadas entre si. Sin embargo, la magnitud misma de esa tarea en la mayoría de las empresas hacen que eso sea imposible, porque las empresas pueden tener miles de empleados, decenas de millares de artículos en inventario, cientos de centros de trabajo, varias plantas y miles de proveedores[1]. Tratar de elaborar cada tres meses planes semanales que abarquen compras, inventario, tasas de producción, niveles de fuerza de trabajo y programas de transporte es un trabajo abrumador para una sola persona o un equipo de personas, si ellos no cuentan con los sistemas de información que funcionen con los métodos y algoritmos adecuados que utilicen las computadoras paralelas actuales.

En el área de administración de suministros se tienen problemas de optimización discreta, por mencionar algunos de ellos: en el área de transporte, la selección de la ruta que nos proporcione el menor costo y que cubra todos los centros de distribución; en la determinación los centros de distribución, donde colocar dichos centros de tal modo que la inversión sea la mínima o que la ganancia sea la máxima; en lo referente a inventarios, que productos son los que se tienen que tener disponibles y cuales son los que aportan mayores ganancias dado el volumen disponible de almacenamiento. Cada uno de estos problemas esta sujeto a restricciones físicas, de tiempo, inversión, cargas fiscales, horarios de los trabajadores, etc.

Además, si suponemos que estos problemas los tiene una empresa de grandes dimensiones, entonces éstos tendrán miles de variables, lo que hace que no sea computable su solución por medio de una búsqueda

[1] *Krajewski Lee J. And Ritzman Larry P. "Administración de Operaciones, estrategia y análisis". Quinta edición. Prentice Hall, México 2000. ISBN: 968-444-411-7. pp 453-490.*

exhaustiva o por métodos tradicionales[2, 3, 4]. Por tanto, es necesario desarrollar nuevos métodos y algoritmos que nos determinen la solución optima exacta utilizando las mejores computadoras paralelas actuales.

Con el propósito de resolver estos problemas de miles de variables se desarrollo el método del árbol de cubos [2, 3, 4]. En la segunda y tercera partes de este capítulo se presentan sus fundamentos y el método. En la cuarta parte los resultados del algoritmo secuencial para solucionar el problema de la selección del subconjunto de objetos con un parámetro. En la quinta parte se presenta el algoritmo paralelo del árbol de cubos, el cual se fundamenta en la primera parte del capítulo, también se presentan los resultados obtenidos con dicho algoritmo en una computadora paralela IBM de 32 procesadores.

2. Método del árbol de cubos.

2.1. Planteamiento del problema.

Sea el conjunto finito $I = \{1, 2, ..., m\}$ y sea el conjunto $B(I)$ el conjunto de todos los subconjuntos, esto es: $B(I) = \{\omega | \omega \subseteq I\}$ y $|B(I)| = 2^m$. Al conjunto $B(I)$ es posible imaginarlo como un cubo de dimensión m, donde los vértices del cubo son subconjuntos $\omega \subset I$, estos subconjuntos tienen un orden parcial entre ellos dado por la teoría de conjuntos con las operaciones: $\subseteq$, $\cup$ y $\cap$.

El cubo de dimensión m lo denotamos como $C(m)$, es claro que los elementos (vértices) del cubo son subconjuntos $\omega \subset I$, es posible escribirlo como el intervalo: $C(m) = [\varnothing, I] = \{\omega \subset I \mid \varnothing \subset \omega \subset I\}$. Igualmente para el intervalo $[\omega_1, \omega_2]$, donde $\omega_1 \subset \omega_2 \subset I$, corresponde a un cubo $C(k)$. $C(k) = [\omega_1, \omega_2] = \{\omega \subset I \mid \omega_1 \subset \omega \subset \omega_2\}$, con la dimensión $k = |\omega_2 \setminus \omega_1|$.

Todos los elementos (vértices) de $C(k)$ son vértices de $C(m)$. Si tenemos dos cubos $C(k_1) = \left[\omega_1^1, \omega_2^1\right]$, $C(k_2) = \left[\omega_1^2, \omega_2^2\right]$ y $\omega_1^2 \subset \omega_1^1, \omega_2^1 \subset \omega_2^2$ entonces se puede escribir $C(k_1) \subset C(k_2)$. Para $k \leq m$: $C(k) \subset C(m)$.

En este trabajo estudiaremos la tarea de partir el cubo $C(m)$ en los conjuntos $\prod$ de cubos de dimensión menor, donde para cada dos cubos la intersección es el conjunto vacío y la unión de todos los cubos es igual a $C(m)$.

Supongamos $\prod = \{C_1, C_2, ..., C_r\}$, donde C_p es un cubo donde $p = 1, 2, ..., r$. Vamos a decir que $\prod$ es la partición sí:

- $C_i \cap C_j = \varnothing$, para cada $i \neq j$, $1 \leq i \leq r$, $1 \leq j \leq r$

[2] *Vladimir Khachaturov, José Crispín Zavala Díaz "Método y algoritmo del árbol de cubos y sus aplicaciones para resolver diferentes problemas de optimización discreta". Memorias del 6º Congreso Nacional en Computación, CIC-IPN, Mayo 2005. Volumen 13, pp 111-126*.

[3] *Vladimir Khachaturov, José Crispín Zavala Díaz "Método y algoritmo del árbol de cubos para resolver problemas de optimización discreta de grandes dimensiones: solución exacta y aproximada". 1st Euro-Latin Worshop on Engineering Systems. Universidad Nacional de Trujillo, Perú. Abril 2005.*

[4] *José Crispín Zavala Díaz, Vladimir Khachaturov "Determinación de los intervalos de estabilidad en la tarea de selección de un subconjunto de objetos con un parámetro fijo con el árbol de cubos". 1st Euro-Latin Worshop on Engineering Systems. Universidad Nacional de Trujillo, Perú. Abril 2005.*

- $\bigcup_{p=1}^{r} C_p = C(m)$

2.2. Procedimiento para la construcción del árbol de cubos

Sea el cubo inicial $C(m) = [\varnothing, I]$, $I = \{1, 2, ..., m\}$, vamos a construir un medio especial con el conjunto parcialmente ordenado de cubos $C(l)$, $l=m-k$ donde $k=0, 1, ..., m$, de diferentes dimensiones. Al nivel superior $l=m$ $(k=0)$ le corresponde el cubo único $C(m)$ de dimensión m. En el nivel siguiente $l=m-1$ $(k=1)$ se distribuyen dos cubos disjuntos: $C_1(m-1)$ y $C_2(m-1)$ de dimensión $(m-1)$ cada uno, estos cubos se pueden hacer de m distintas formas, nosotros hacemos lo siguiente: para el cubo $C_1(m-1)$ consideramos un intervalo $[\{1\}, I]$ y al cubo $C_2(m-1)$ le corresponde un intervalo $[\varnothing, I\backslash\{1\}]$. Estos dos cubos son subconjuntos disjuntos, $C_1(m-1) \cap C_2(m-1) = \varnothing$, porque todos los elementos del cubo $C_1(m-1)$ tienen al subconjunto $\{1\}$ y todos los elementos del cubo $C_2(m-1)$ no tienen al mismo elemento $\{1\}$.

El número de elementos de cada uno de los cubos C_1 y C_2 es igual a $2^{(m-1)}$, entonces el número de elementos de ambos cubos es 2^m, por eso: $C_1(m-1) \cup C_2(m-1) = C(m)$. Por tanto, el conjunto de los dos cubos $C_1(m-1)$ y $C_2(m-1)$ es la partición del cubo $C(m)$. Notamos que cada cubo del nivel $l=m-k$ tiene la dimensión $l=m-k$, entonces tiene 2^l vértices.

Posteriormente se forman los niveles siguientes $(m-2)$, en cada cubo $C_1(m-1)$ y $C_2(m-1)$ se hacen las mismas operaciones. Cada cubo se parte en dos cubos de una dimensión menor, entonces en el nivel $(m-2)$ recibimos a cuatro que son una partición del cubo $C(m)$. Si seguimos este proceso llegamos al nivel $m-3$ con ocho cubos, dos cubos del nivel $m-3$ para cada uno de los cubos del nivel $m-2$. En general para el nivel $(m-k)$, $0 \le k \le m$, se pueden formar 2^k cubos, donde cada dos cubos son conjuntos disjuntos y la unión de todos ellos forman el cubo inicial $C(m)$. Entonces todos los cubos del nivel $(m-k)$ es la partición del cubo $C(m)$.

En el último nivel $(l=0)$ el número de cubos de dimensión cero es igual a 2^m. Entonces es igual al número de vértices del cubo inicial $C(m)$. El conjunto de ellos es la partición, porque es claro que cada intersección es igual a cero y la unión de ellos es el conjunto inicial $C(m)$.

Construimos un árbol donde sus vértices son los cubos definidos con el procedimiento descrito. En cada etapa donde recibimos cubos, en el nivel siguiente se construye un lado entre el cubo del cual nacieron con los dos cubos de una dimensión menor en uno del cubo del nivel superior. Como resultado recibimos un árbol con vértices, donde cada vértice es un cubo y ellos son de un orden parcial definido por el lado que los conectan. Supongamos este árbol como $A(m)$.

2.3 Algunas propiedades del árbol de cubos

2.3.1. **Estructura jerárquica.**

De la formalización del algoritmo de la construcción del árbol de cubos concluimos que el árbol tiene una estructura jerárquica. El árbol tiene $m+1$ niveles $l = m - k$ $(k = 0, 1, ..., m-1, m)$, en el nivel superior $(l=m)$ está un sólo vértice, en los niveles siguientes el número de vértices (número de cubos) es igual a 2^k $(k = 0, 1, ..., m-1, m)$.

2.3.2. **Número de vértices del árbol *A(m)***

El número de todos los vértices del árbol $A(m)$ es igual a: $1 + 2 + 4 + ... + 2^m$. Entonces $|A(m)| = 2^{m+1} - 1$.

2.3.3 Generación de diferentes variantes de particiones del árbol *A(m)*

El conjunto de vértices de cada nivel del árbol *A(m)* es la partición del cubo inicial *C(m).* Entonces en cada árbol se tiene al mismo tiempo *m* diferentes variantes de las particiones del cubo inicial *C(m).* Además se puede formar muchas otras variantes de la partición. Por ejemplo se puede tomar como base una variante de la dimensión que le corresponde a cualquier nivel *(l > 0)* del árbol y sustituir, dentro de esta variante base, uno de los elementos (o varios elementos) con dos cubos de dimensión menor a uno, que nosotros recibimos con el procedimiento de construcción del árbol. Esta propiedad del árbol de cubos permite organizar eficazmente los cálculos en las máquinas paralelas.

2.3.4. Dependencia del número de vértices del árbol conforme a su nivel

La dependencia del número de vértices del árbol *A(m)* conforme al nivel *l (N(l), l=m-k,* donde *k = 0, 1, 2, ..., m)* ésta esta dada por:

$$N(k) = 2|x(l(k))| = 2^{m-l(k)} = 2^k$$

Donde

$l(k)=m-k$, $(k = m, m-1, ..., 1, 0)$ es el número del nivel

$|x(l(k))|=2^{m-1-l(k)}=2^{k-1}$

Si reflejamos esta función (dependencia) *N(l)* en los sistemas coordenados *(x, l)*, generamos una gráfica, como se muestra en la figura *1*. Esta figura recuerda el contorno de un pino que tiene un tronco que corresponde a una franja $-\frac{1}{2} \le x \le \frac{1}{2}$. El contorno de este pino se compone de dos aproximaciones $x(l(k))=2^{k-1}$ para $x(l(k)) \ge 0$ y $x(l(k)) = -2^{k-1}$ para $x(l(k)) \le 0$. Por eso $N(x(l(k)))=2^{k-1}-(-2^{k-1})=2^k$. Notamos que si aumentamos la magnitud de *m*, el "pino" se desliza a lo largo del eje "*l*". El incremento de la capacidad de vértices del cubo sucede en los últimos niveles del "pino" nuevo. En la figura *2* el árbol de cubos para $m = 5$.

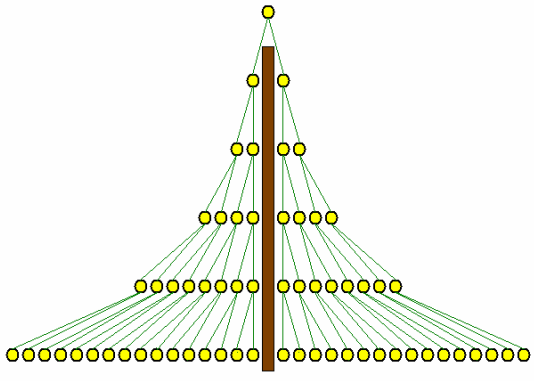

Figura 1. El árbol de cubos para $m = 5$. En el centro el tronco va de $-\frac{1}{2} \le x \le \frac{1}{2}$

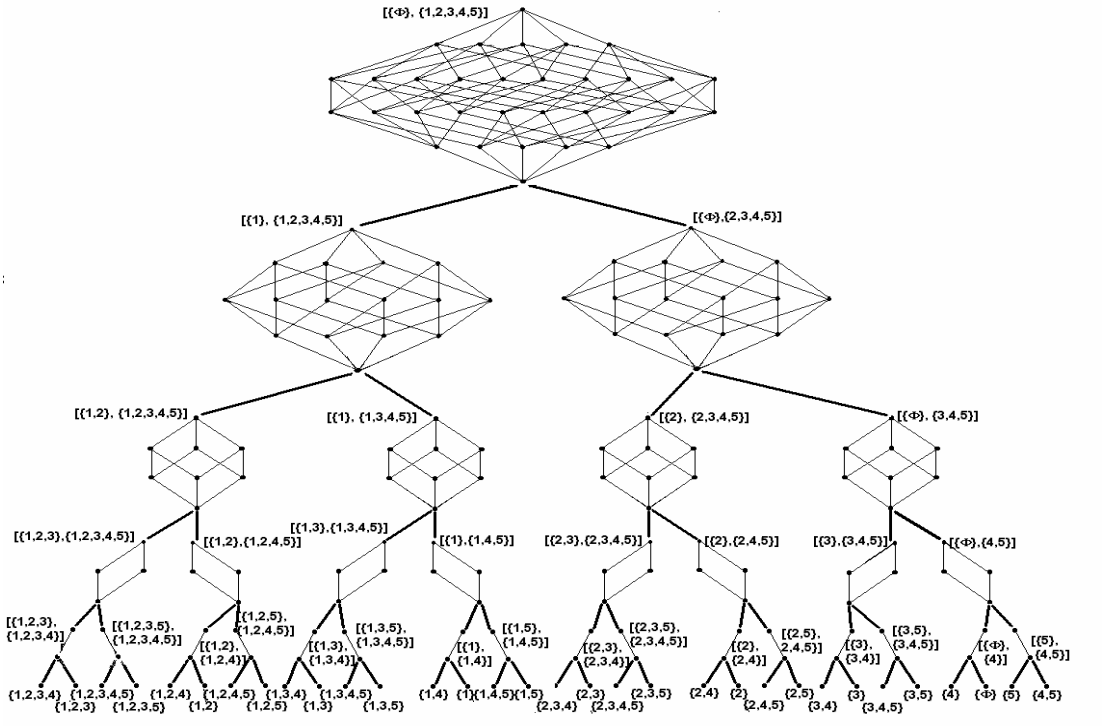

Figura 2 El árbol de cubos

2.3.5. Número de vértices como una función de los intervalos de los niveles del árbol

No es difícil determinar, para dos pinos con $m=m_1$ y $m=m_2$ el número de vértices del árbol de cubos en los intervalos $[m_1\text{-}k, m_1]$ y $[m_2\text{-}k, m_2]$ es el mismo, para cualquier $0 \le k \le min(m_1, m_2)$ y es igual a $2^{k+1}-1$, entonces no depende del valor de m

2.4. Subárboles del árbol *A(m)* y sus propiedades.

Cada vértice del árbol forma un subárbol del árbol *A(m).* El vértice del árbol *A(m)* – cubo *C(m)* - se denomina también la raíz del árbol *A(m),* porque de él nacen todos los vértices del árbol.

Un subárbol del árbol *A(m)* es un árbol que nació a través de cualquier vértice del árbol *A(m).* El número de todos los subárboles es igual al número de todos los vértices del árbol, esto es igual a $2^{m+1}\text{-}1$, cada subárbol que nace de la raíz *C(m-k)* tiene $2^{m-k+1}-1$ vértices *(k=0, 1, ..., m).*

Entonces para $k=0$ tenemos la raíz del árbol principal *A(m)* que tiene $2^{m+1}-1$ vértices. Para $k=m$ cada uno de los 2^m subárboles tienen un solo vértice.

2.4.1. Sea $C^1(l_1)$ y $C^2(l_2)$, $l_2 \le l_1$, dos raíces que corresponden a dos subárboles $A^1(l_1)$ y $A^2(l_2)$.

- Sí $C^1(l_1) = C^2(l_2)$, entonces $A^1(l_1) = A^2(l_2)$.
- Si $C^1(l_1) \cap C^2(l_2) = \varnothing$, entonces $A^1(l_1) \cap A^2(l_2) = \varnothing$, que significa que los subárboles $A^1(l_1)$ y $A^2(l_2)$ no tienen vértices y lados comunes.

- Si $C^2(l_2) \subset C^1(l_1)$, entonces todos los vértices y lados del subárbol $A^2(l_2)$ también son elementos del subárbol $A^1(l_1)$.

2.4.2. Si excluimos del árbol $A(m)$ el subárbol $A(l_0)$, $l_0<m$, excluiremos a todos los subárboles $A(l)$ con $l<l_0$, para los cuales $C(l) \subset C(l_0)$.

2.4.3. Si alguna propiedad no cumple para ningún elemento $\omega \in C(l_0)$, entonces esta propiedad no cumple para ningún elemento $\omega \in C(l)$ para todo $C(l) \subset C(l_0)$, porque $C(l_0)$ tiene todos los elementos para cualquier $C(l) \subset C(l_0)$.

2.4.4. Si alguna propiedad cumple para algún elemento $\omega_0 \in C(l_0)$, entonces existe una cadena de cubos: $C^0(0) \subset C^0(1) \subset C^0(2) \ldots C^0(l_0-1) \subset C^0(l_0)$, en la cual para cada $C^0(l)$, $0 \le l \le l_0$, existe un elemento donde esta propiedad también se cumple. Por ejemplo, la cadena de cubos para los cuales $C^0(0) = \omega_0$

2.5. El árbol de cubos como latices

Introducimos en el árbol de cubos un vértice, el conjunto vacío, y lo conectamos por medio de lados con todos los cubos del nivel cero $(l(k)=0, k=m)$. Por el teorema siguiente determinamos que la estructura que recibimos es un latice [5, 6].

Teorema. Después de introducir en el árbol de cubos el vértice que corresponde al conjunto vacío, la estructura que recibimos es una latice.

Demostración. En correspondencia con la formación del árbol de cubos m, cada dos cubos $C_1 = \left[\omega_1^1, \omega_2^1\right]$, $C_2 = \left[\omega_1^2, \omega_2^2\right]$ y $C_1 \neq C_2$ pueden tener una de las dos propiedades:

1. $C_1 \subset C_2$, $\omega_1^2 \subset \omega_1^1$ y $\omega_2^2 \supset \omega_2^1$
2. $C_1 \cap C_2 = \varnothing$

Por el primer punto se tiene: supremo $(C_1, C_2) = C_2$, ínfimo $(C_1, C_2) = C_1$.

Por el segundo punto se tiene: el supremo (C_1, C_2) es un nuevo cubo $\tilde{C} = \left[\omega_1^1 \cap \omega_1^2, \omega_2^1 \cup \omega_2^2\right]$ sí el cubo $\tilde{C} \in A(m)$, sí $\tilde{C} \notin A(m)$ supremo $(C_1, C_2) = C(m) = [\varnothing, I]$ e ínfimo $(C_1, C_2) = \varnothing$. Entonces nosotros determinamos que después de introducir el conjunto vació en el árbol de cubos m, nosotros recibimos la estructura en la cual cualquier par de elementos siempre tienen supremo e ínfimo, por eso la estructura que recibimos es una latice [5,6].

3. Método del árbol de cubos para los problemas de optimización

[5] *Birkhoff G., "Lattice Theory", second edition Amer., Math Soc., Providence, 1948*

[6] *Grätzer G. "General Lattice Theory" Akademie-Verlag Berlin, 1978.*

Con el objetivo de explicar cómo el método de árbol de cubos se aplica para determinar la solución de problemas de optimización discreta, utilizamos el planteamiento del problema de la selección optimal del subconjunto de objetos con un parámetro, que es una generalización de la tarea clásica conocida como “la tarea del maletín”.

3.1 Planteamiento del problema de la selección optimal del subconjunto de objetos con un parámetro.

Se tiene un conjunto de *m* objetos *I = {1, 2, ..., m}*, cada objeto $i \in I$ tiene un peso $b_i > 0$ y un valor $c_i > 0$. La función de optimización esta dada por:

Es necesario buscar: $\max \sum_{i=1}^{m} c_i x_i$

Sujeto a: $\sum_{j=1}^{m} b_i x_i \le \lambda B, \quad x_i \in \{0,1\}$,

Donde: $B = \sum_{i=1}^{m} b_i, \quad 0 \le \lambda \le 1$

3.2 Algoritmo para la realización del método de árbol de cubos – Algoritmo de Árbol de Cubos (*AAC*)

Es claro que si:

- $\lambda=0$ la solución de la tarea es *X(0, 0, ..., 0)* y el máximo $\sum_{i=1}^{m} c_i x_i = 0$
- $\lambda=1$ la solución de la tarea es *X(1, 1, ..., 1)* y el máximo $\sum_{i=1}^{m} c_i x_i = \sum_{i=1}^{m} c_i$

Vamos a describir el algoritmo para cualquier valor de λ para $0<\lambda<1$. Se selecciona una λ_0, $0<\lambda_0<1$, y se calcula $b = \lambda_0 B$.

3.2.1 Procedimiento para determinar la solución aproximada

Se obtienen los coeficientes: $\frac{c_1}{b_1}, \frac{c_2}{b_2}, ..., \frac{c_m}{b_m}$ y los ordenamos de mayor a menor: $\frac{c_{i1}}{b_{i1}} \ge \frac{c_{i2}}{b_{i2}} \ge ... \ge \frac{c_{im}}{b_{im}}$. Expresamos con estas variables la función objetivo y la restricción.

$$\max \sum_{j=1}^{m} c_{i_j} x_{i_j}$$

Sujeto a: $\sum_{j=1}^{m} b_{i_j} x_{i_j} \le b$, $x_{ik} \in \{0,1\}$,

tal que $i_k \in I$ para $k = 1, 2, ..., m$, $i_k \ne i_l$ para $k \ne l$

Transformando estas variables con $y_k = b_{ik} x_{ik}$, obtenemos:

$$\max \sum_{j=1}^{m} \frac{c_{i_j}}{b_{i_j}} y_j$$

Sujeto a: $\sum_{j=1}^{m} y_j \le b$, $y_k \in \{0, b_{ik}\}$

De esta última transformación se obtienen las soluciones integral *C(I)* y lineal *C(L)*. Dado que b_{i_k} es un entero, se tendrá un límite *q* tal que:

$$\sum_{k=1}^{q} b_{ik} \le b < \sum_{k=1}^{q+1} b_{ik}$$

Porque $\sum_{k=1}^{m} b_{i_k} = B > \lambda_0 B = b$ entonces $q < m$ siempre.

Con lo cual se obtienen los elementos que son la solución.

$y_k = b_{i_k}$, cuando $x_{i_k} = 1$ para *k = 1, 2, ..., q*

$y_k = 0$, cuando $x_{i_k} = 0$ para *k = q+1, ..., m*

La magnitud de la función objetivo para la solución integral estará dada por los términos donde $x_{i_k} = 1$ ó $x_{i_k} = 0$

$$C(I) = \sum_{k=1}^{q} c_{i_k}$$

Para la solución lineal (no integral) tenemos:

$$C(L) = C(I) + \frac{c_{i_{(q+1)}}}{b_{i_{(q+1)}}} \left(b - \sum_{k=1}^{q} b_{i_k} \right)$$

Claro que *C(L)≥C(I).* Puede ser *C(L)=C(I)* sólo cuando $b = \sum_{k=1}^{q} b_{i_k}$. A cada solución integral, $x_{i_k} = 1$, le corresponde el subconjunto $\omega^0 \subset I$, donde $i_k \in \omega^0$ para $x_{i_k} = 1$ y $i_k \notin \omega^0$ para $x_{i_k} = 0$. A cada ω^0 le corresponde su *C(ω⁰),* donde $C\left(I, \omega^0\right) = \sum_{i \in \omega^0} c_i$ y *C(L, ω⁰)≥C(I, ω⁰).*

Este algoritmo se puede utilizar para cualquier subárbol *A(l)* con su raíz *C(l)* de la manera siguiente. Cada una de las raíces se puede representar de la forma:

C(l) = (ω₁, ω₂) = {ω|ω₁⊂ ω⊂ω₂}, donde *[ω₂\ω₁] = l, ω₁⊂I* y *ω₂⊂I.*

Planteamiento del problema, para el cubo *C(l)* es necesario buscar:

$$\sum_{i \in \omega_1} c_i + \max \sum_{i \in \omega_2 \setminus \omega_1} c_i x_i$$

Sujeto a: $\sum_{i \in \omega_2 \setminus \omega_1} b_i x_i \le b - \sum_{i \in \omega_1} b_i x_i$, $x_i \in \{0, 1\}$, $i \in \omega_2 \setminus \omega_1$

donde: $x_i = 1$ para $i \in \omega_1$,

$x_i = 0$ para $i \in I \setminus \omega_2$

Buscamos $\max \sum_{i \in \omega_2 \setminus \omega_1} c_i x_i$ igual que para el cubo *C(m).* Como resultado recibimos, para el cubo *C(l)=C(ω₁,ω₂),* las soluciones optimales lineal *C(L,ω⁰,l)* e integral aproximado con magnitud *C(I,ω⁰, l).* Claro, siempre *C(L,ω⁰,l) ≥ C(I,ω⁰, l)* para *ω₁⊂ω⁰⊂ω₂.*

3.2.2 Reglas para el rechazo de subárboles no optimales.

En correspondencia con la propiedad 2.4.3, si en el cubo $C(l_0)$ no existen soluciones optimales, entonces no hay soluciones optimales en ningún cubo $C(l) \subset C(l_0)$. Por tanto si de alguna manera afirmamos que el cubo $C(l_0)$ no tiene soluciones optimales, entonces no hace falta revisar todo el subárbol $A(l_0)$, en consecuencia no hace falta revisar los $2^{l_0+1} - 1$ cubos, vértices de este subárbol.

Para el problema que investigamos se pueden formular las siguientes reglas de rechazo de subárboles no optimales.

Supongamos conocidas algunas soluciones aproximadas de nuestra tarea, las cuales están en el subconjunto $\tilde{\omega} \subset I$ con sus correspondientes magnitudes de las funciones de la solución integral $C\left(I, \tilde{\omega}\right)$ y de la solución no integral (lineal) $C\left(L, \tilde{\omega}\right)$, donde $C\left(I, \tilde{\omega}\right) \le C\left(L, \tilde{\omega}\right)$

Sea el subárbol *A(l)* con su raíz $C(l)=C(\omega_1,\omega_2)$, para *C(l)* conocemos las magnitudes de sus soluciones $C(L,\omega^0, l)$ y $C(I,\omega^0, l)$, donde $C(L,\omega^0, l) \geq C(I,\omega^0, l)$.

Regla de rechazo

Si $C(I,\tilde{\omega}) \geq C(L,\omega^0,l)$, entonces se puede excluir la revisión del subárbol *A(l)*, porque entre todos los cubos que son vértices del subárbol *A(l)* no existe ninguna solución "mejor" que $\tilde{\omega}$, es decir, no existe una solución con una significación del funcional entero mayor que $C(I,\tilde{\omega})$. La demostración de esta regla sigue la desigualdad siguiente:

$$C(I,\tilde{\omega}) \geq C(L,\omega^0,l) \geq C(I,\omega^0,l)$$

y de la propiedad de los subárboles 2.4.3.

Esta regla se utiliza permanentemente en el proceso de búsqueda de la solución optimal de la tarea inicial.

3.2.3 Algoritmo de búsqueda de la solución optimal

El algoritmo de búsqueda de las soluciones optimales se constituye de cuatro bloques principales (etapas).

Primer Bloque. Diferentes búsquedas de las soluciones integrales aproximadas. La realización de este bloque es una etapa muy importante en la búsqueda de la solución optimal, porque la presencia de una buena solución integral permite aumentar la eficacia del algoritmo de búsqueda de la solución optimal, ya que aumenta la eficacia de la regla de rechazo.

Como resultados de la realización de este bloque determinamos y guardamos en la memoria la solución integral $\tilde{\omega}$ y su correspondiente significación $C(\tilde{\omega})$, la que determinamos como la solución optimal temporal.

Segundo Bloque. Esquema de selección de los vértices del árbol *A(m)*. Existen diferentes algoritmos para seleccionar los vértices de un árbol, por ejemplo las que se presentan en las referencias[7, 8].

Nosotros utilizamos los algoritmos, que se han aplicado durante muchos años, para solucionar diferentes tareas de optimización combinatoria [9]. El esquema común esta escrito en el punto 2 del presente trabajo.

Tercer Bloque. Realización de la regla de rechazo. Para cada vértice $C(l)=C(\omega_1,\omega_2)$ determinamos las significaciones $C(L,\omega^0,l)$, $C(I,\omega^0, l)$ y la solución integral aproximada ω^0 , $\omega_1 \subset \omega^0 \subset \omega_2$. Después comparamos $C(L,\omega^0,l)$ con la significación del óptimo temporal $C(I,\tilde{\omega})$.

---

[7] *Nemhauser G. L., Wolsey L. A. "Integer and Combinatorial Optimization" A Wiley-Interscience Publication, John Wiley and Sons Inc., New York 1999*

[8] *Pisinger D., ``An exact algorithm for large multiple knapsack problems'',* European Journal of Operational Research*, 114 528-541 (1999)*

[9] *Khachaturov V. R. "The modeling and methods of solutions of multiextremal allocations problems and the use of supermodular function proprieties in the Boolean lattices", Algorithms and Languages of algorithms, Moscu: "Nauka", 1986*

Si $C(I,\tilde{\omega})$≥*C(L,*ω^0*,l)* excluimos del cálculo al subárbol *A(l),* entonces no continuamos formando dos nuevos cubos del vértice *C(l),* y pasamos a formar nuevos cubos de otro vértice en correspondencia al esquema común de formalización del árbol *A(m)* (ver punto dos). Si no existen otros cubos terminamos la búsqueda de la solución optimal y pasamos al bloque 4.

Cuarto bloque. Retención de las soluciones optimas temporales y la salida del cálculo. Cada vez, después de la determinación de la solución integral, para cada vértice *C(l)* del árbol *A(m)* la significación de *C(I,*ω^0*,l)* se compara con la significación $C(I,\tilde{\omega})$. Si *C(I,*ω^0*,l)* > $C(I,\tilde{\omega})$, entonces ω_0 con su *C(I,*ω^0*,l)* la retenemos como la solución siguiente optimal temporal en lugar de $\tilde{\omega}$.

Después de concluir el cálculo, la última solución optimal temporal $\tilde{\omega}_u, C(I,\tilde{\omega}_u)$es la solución de la tarea inicial $\omega_{opt} = \tilde{\omega}_u, C(I,\omega_{opt}) = C(I,\tilde{\omega}_u)$ son las salidas del algoritmo.

4. Los experimentos computacionales, versión secuencial

Los experimentos computacionales consistieron en obtener la solución integral, lineal y los subconjuntos de la solución optimal para cada λ_0 seleccionada. Se consideraron dos tipos de problemas, el primero de ellos fue determinar los coeficientes b_i y c_i por medio de una progresión aritmética, de tal forma que nosotros conocemos la suma de ellos para cualquier número de elementos, es decir la solución optimal. Una vez que se determinaron los coeficientes, éstos se distribuyeron en diferentes posiciones para dirigir la búsqueda a cubos diferentes. En todas las pruebas, el AAC determinó la solución optima correcta, con lo cual comprobamos que el algoritmo del árbol de cubos se implementó correctamente.

El segundo experimento consistió en determinar los coeficientes *(b_i, c_i)* en forma aleatoria, los resultados que se presentan son para este segundo experimento. El primer experimento que se presenta para *m=500,* consiste en determinar la solución entera optima para *20* intervalos de λ. Para este problema y el siguiente se utilizó la regla de rechazo general *C(L,*ω^0*,l)*- $C(I,\tilde{\omega}) \le \varepsilon C(I,\tilde{\omega})$. Cuando $\varepsilon = 0$ la regla de rechazo se utiliza para determinar la solución exacta, y cuando ε > 0 buscamos la solución aproximada con un valor no mayor a $\varepsilon C(I,\tilde{\omega})$. En la figura 3 se muestra la gráfica de la solución exacta, donde en las abscisas está λ y en el eje de las ordenadas está el número de iteraciones con el cual convergió el algoritmo.

Como es de suponerse, los cálculos experimentales confirman nuestros supuestos de que los problemas mas difíciles requieren de muchos cálculos, esto es cuando $\lambda \approx 1/2$.

El número de soluciones *(N)* que calculamos nunca fue mayor a m^4, $N<m^4$. En forma indirecta nos muestra que el algoritmo que utilizamos es efectivo. Tomando como base estos valores experimentales ($N<m^4$, para $\lambda=1/2$), nosotros recibimos en forma analítica los valores para otras significaciones de λ siguientes:

$N \approx m^{x_k}$, donde $x_k = 4\left(1-\log_m 2^k\right)$, *k = 0, 1, 2, ...*

para $0 < \lambda_k \le \frac{1}{2}$, donde $\lambda_k = 2^{-(k+1)}$ y

para $\frac{1}{2} \le \lambda_k < 1$, donde $\lambda_k = 1-2^{-(k+1)}$.

Es claro que para $\lambda_k = 0$ la solución optimal es $x_i = 0$ para todos *i = 1, 2, ..., m,* y para $\lambda_k = 1$ la solución optimal es $x_i = 1$ para todos *i = 1, 2, ..., m.*

La magnitud de la solución optimal entera se muestra en la figura 4, como se observa la magnitud de la solución entera se incrementa monótonamente hasta alcanzar el máximo, cuando $\lambda=1$.

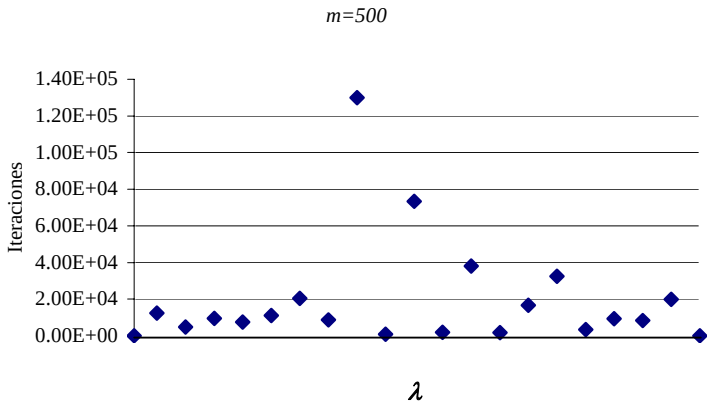

Figura 3. Iteraciones en función de λ para $m = 500$.

El segundo experimento que se presenta es para $m= 3{,}200$ con diferentes valores de ε. Se determinan las soluciones aproximadas para $\varepsilon>0$ *(6%, 4%, 2%* y *1%)* y la solución exacta $\varepsilon=0$. El calculo se realizó para $\lambda = 0.0035$ a causa de que es posible obtener la solución exacta en un procesador, ya que como se mostró en el primer ejemplo, para una λ mayor y cercana a ½ se requiere de un número de iteraciones considerablemente alto, pero nunca mayor a m^4. Los resultados de este segundo ejemplo se muestran en la tabla 1.

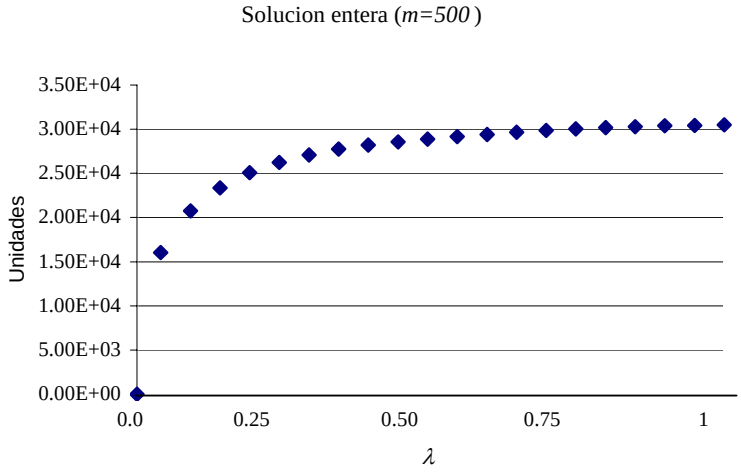

Figura 4. Magnitud de la solución entera exacta para $m=500$

En la segunda columna de la tabla 1 están los números de iteraciones en que se calculo el valor mayor y en la tercera columna está el número de iteraciones en que se determino que este valor máximo era la solución optima. En la cuarta columna está el valor de la solución entera y en la quinta columna está el intervalo en el cual la solución optima se encuentra.

El intervalo de la solución estará dado por el porcentaje considerado para obtener la solución aproximada ($\varepsilon C(I,\tilde{\omega})$). Por ejemplo para ε=6% el límite superior del intervalo es *2,202 + 0.06(2,202)= 2,334,* el cual está en la quinta columna de la tabla 1. Para el porcentaje siguiente el límite superior del intervalo es *2,401,* pero dado que la solución no será mayor al límite superior anterior, entonces consideramos éste como el límite superior del intervalo. Los resultados para los demás porcentajes se muestran en la quinta columna.

En esa tabla 1 se observa que en todos los casos la solución exacta está dentro de los intervalos calculados de las soluciones aproximadas. El costo de obtener la solución exacta se ve reflejada en el número de iteraciones requeridas para determinar el valor máximo y para determinar que este máximo es la solución optima.

Tabla 1. Solución aproximada y exacta para *m=3,200* con una λ = *0.0035%*

ε(%)	***Iteración máximo***	***Iteración optimo***	$C(I)_{aprox}$	***C(I)***
6	*4*	*5*	*2,202*	$2{,}202 \leq C(I)_{opt} \leq 2{,}334$
4	*62*	*1339*	*2,309*	$2{,}309 \leq C(I)_{opt} \leq 2{,}334$
2	*338*	*1341*	*2,309*	$2{,}309 \leq C(I)_{opt} \leq 2{,}334$
1	*1338*	*1343*	*2,309*	$2{,}309 \leq C(I)_{opt} \leq 2{,}332$
0 (exacta)	*1378*	*2655*	*2,329*	$C(I)_{opt} = C(I)_{aprox} = 2{,}329$

A causa de la diferencia tan grande de iteraciones, entre la solución exacta y la aproximada, para este problema (*m=3,200)* es posible calcular las soluciones enteras aproximadas para cualquier λ siempre y cuando $\varepsilon \geq 1\%$**.** Para calcular la solución optima exacta de cualquier λ se necesita de cómputo paralelo por la gran cantidad de cálculos y memoria requeridos.

5. Árbol de cubos paralelo

La solución paralela de cualquier tipo de problemas se puede hacer utilizando:

- Paralelismo a través del sistema o del espacio.
- Paralelismo a través del método o paralelismo a través del tiempo

En el primer grupo se incluyen los algoritmos que utilizan el espacio del problema y no el método de solución. Mientras que en el segundo grupo se incluyen los algoritmos que utilizan las funciones concurrentes simultáneas[10], como es el método del árbol de cubos.

Como se describió previamente, el método inicia con cubos de mayor dimensión y conforme el cálculo avanza los cubos son de menores dimensiones, ya que están en niveles inferiores. Donde los cubos generados no persisten durante toda la búsqueda de la solución optima, porque al enviar la información a sus cubos hijos los cubos padres dejan de ser necesarios. Al aplicar la regla de rechazo se seleccionan los cubos sobre los cuales se continuará el cálculo, conocer a priori los cubos sobre los que se seguirá el cálculo no es posible por el proceso

[10] *K. R. Jackson and S. P. Norsett "The potential for Parallelism in Runge-Kutta methods. Part 1: RK Formulas in Standar Form". Computer Science Department, University of Toronto, Canada, November 1990.*

mismo y del problema. Por tanto se requiere diseñar un algoritmo de distribución de cubos que considere lo siguiente:

a) Cada cubo se puede calcular en forma independiente de otros cubos que sean conjuntos disjuntos con él, porque no comparten elementos comunes
b) La información que le envía el cubo padre a cada uno de sus dos cubos hijos son sus respectivos conjuntos ω_1, ω_2 y la mejor solución entera encontrada hasta esa iteración.
c) Durante el proceso de cálculo sólo se requiere de dos puntos de sincronización para enviar y recibir información. El primero es cuando el Cubo recibe la información y cuando este, a su vez, le envía la información a sus dos cubos hijos.
d) El árbol de cubos lo podríamos clasificar como un árbol del tipo *OR*[11], donde no es necesario calcular todas las ramas del árbol.
e) La arquitectura de la computadora paralela donde se implantará el algoritmo paralelo.

5.1 Determinación de la configuración de la computadora paralela

La computadora paralela disponible es de configuración dinámica, tiene 32 procesadores agrupados en grupos de cuatro, donde estos cuatro procesadores se comunican por medio de una memoria compartida con acceso por medio de bus, lo grupos se comunican a través de una memoria compartida por medio de otro bus[12], con lo cual se tiene comunicación punto a punto. Por tanto, recordando que el método del árbol de cubos es un árbol binario, la arquitectura considerada es isomorfa al método del árbol de cubos, donde en cada procesador se determina la solución optima de un cubo a la vez. Posteriormente, los cubos hijos se asignan a otros procesadores y así sucesivamente hasta determinar el óptimo. Cuando el procesador envía los subconjuntos a los procesadores donde están los hijos, este procesador queda disponible para calcular otros cubos.

Las conexiones entre los procesadores para la distribución de los cubos se realiza considerando lo siguiente:

1. El número de cada cubo en el árbol es diferente y dependiente del número del cubo padre. Si *j* es el número del cubo padre, el hijo izquierdo tendrá el número *2*j+1* y el hijo derecho *2*j+2*. Con esto, cada nuevo cubo tiene un número distinto e irrepetible.
2. Si *i* es el número de cubo generado y *p* el número de procesadores disponibles, entonces el procesador que recibe al cubo es $q = i \bmod p$.
3. En el caso de que dos o mas cubos sean asignados al mismo procesador, es posible aplicar nuevamente la regla de rechazo, de tal forma que se pueda hacer una nueva eliminación de cubos y reducir aún mas el proceso de cálculo.

Considerando los puntos 1 y 2, la arquitectura resultante se muestra en la figura 5 para 31 procesadores. Donde cada circulo representa un procesador y las flechas que los unen son las conexiones entre ellos, el sentido de la

[11] *Michael J. Quinn "Parallel Computing Theory and Practice, second edition". Ed. McGraw Hill. Inc. 1994, pp 337-338.*

[12] *Manual de la computadora IBM Regatta*

flecha indica el procesador que envía los subconjuntos. Cuando los cubos asignados en los procesadores que están en el último nivel necesitan enviar cubos hijos, éstos se asigna a los procesadores que están en los niveles superiores del árbol de la arquitectura de la computadora, determinados por la función de módulo (punto 2.), lo cual equivaldría a colocar un subárbol cada vez que sea necesario, esto nos proporciona una arquitectura de computadora capaz de calcular árboles de cualquier tamaño. Por ejemplo el procesador 15 envía a los procesadores 0 y 1, cuando envía al procesador 0 equivale a ponerle otra arquitectura de 31 procesadores.

Con esta configuración es posible que en cada subárbol se determine una solución máxima o mínima, según sea el caso, lo que hace necesario que en forma regular se comparen esas soluciones y se obtenga la solución optima.

Esta configuración se puede aplicar a cualquier número de procesadores siempre y cuado sumen $2^n - 1$. donde $n \in Z^+$.

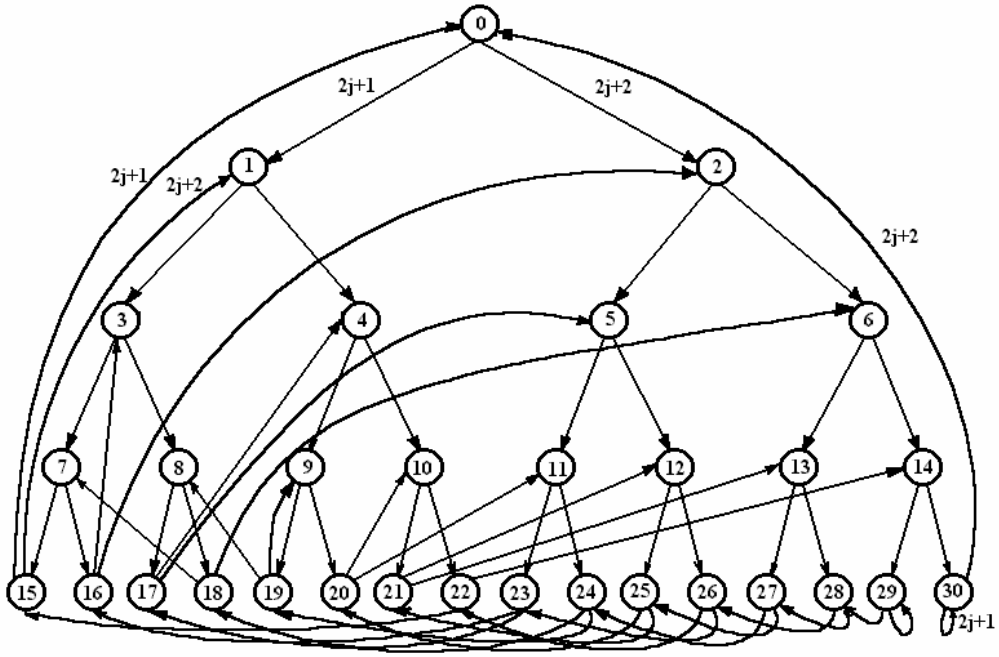

Figura 5. Configuración de la computadora paralela con 31 procesadores

5.2. Algoritmo paralelo del árbol de cubos.

Conocido el método de árbol de cubos y la configuración de la computadora paralela, el algoritmo paralelo tiene que ser eficiente, donde para lograr un buen balance de carga es conveniente que los procesadores calculen el mismo número de cubos de los mismos niveles. Además de considerar reducir al mínimo los tiempos utilizados para [13]:

[13] *José Crispín Zavala Díaz "Una aportación metodológica para solucionar en forma paralela un problema numérico complejo utilizando la variante explícita" Tesis de doctorado. ITESM Campus Cuernavaca, noviembre de 1999.*

- El asociado con la transferencia de los mensajes *(T_{comm}).* Incluye los tiempos en que inicia y termina el envío de un mensaje
- El tiempo asociado con las operaciones locales *(T_{local}).* Los "sobrecargos" locales de la calendarización dinámica, cuando al procesador *p* se le asignan más de un módulo.
- El tiempo de espera *(T_{wait}).* Es el tiempo en que el procesador utiliza en completar todos sus módulos, en esperar los mensajes en tránsito y en esperar el envío de un mensaje.

El algoritmo paralelo del árbol de cubos se muestra en la figura 6, se muestra en la máquina de estados finitos los estados y las transiciones, donde las transiciones son los subprogramas y los estados indican las entradas y salidas de cada uno de ellos. En la tabla 2 se indica que subprogramas son ejecutados en paralelo. El algoritmo se codifico en C con la librería de paso de mensajes MPI [14]

5.3. Experimentos computacionales del algoritmo paralelo

En esta parte del capítulo presentamos nuestros primeros resultados obtenidos con el algoritmo paralelo. Cuya meta es probar que la implantación paralela es correcta, lo cual se obtiene calculando la misma solución optima que se obtiene del algoritmo secuencial.

El problema seleccionado fue el que se presentó previamente para $m = 500$ y una $\lambda = 0.452$, el motivo de seleccionar una λ cercana a 0.5 fue tomar una sección donde el cómputo es mas intensivo, como se muestra en la figura 3.

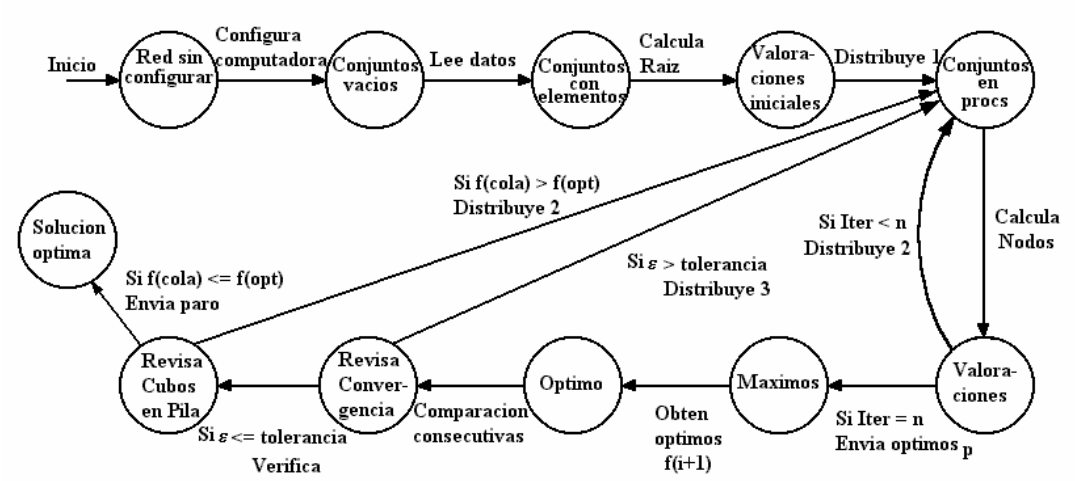

Figura 6. Máquina de estados finitos del algoritmo paralelo del árbol de cubos

[14] *Peter S. Pacheco "Parallel Programming with MPI", Morgan Kaufmann Publishers, Inc, 1997, San Francisco California.*

Tabla 2. Breve descripción de los subprogramas del algoritmo paralelo

Procesador	Subprograma	Comentarios
Todos	Configura computadora	Cada procesador determina de quien recibe los mensajes y a quien le envía, figura 5.
0	Lee Datos	
0	Calcula Raíz	
0	Distribuye 1	El procesador "*0*" envía a todos los procesadores los coeficientes de la función objetivo y la restricción. Envía los conjuntos ω_1, ω_2 y valoraciones.
Todos	Calcula Nodos	Calculan el cubo asignado
Todos	Distribuye 2	Envía los conjuntos ω_1, ω_2 y valoraciones.
Todos	Envía optimos	Se envían las mejores valoraciones determinadas en cada procesador
0	Obten optimos	Se determina el óptimo
0	Comparación consecutivas	Se verifica convergencia requerida
Todos	Verifica	Verifica que en la cola de cubos de cada procesador no exista otro con una mejor solución que la calculada
0	Distribuye 3	Envía la mejor solución a los procesadores y los datos para continuar el cálculo
0	Envía paro	Detiene el cálculo, la solución optima ha sido encontrada

La ejecución del algoritmo secuencial se realizó en una computadora personal con un procesador Intel Pentium IV a 2.4 Ghz y con un sistema operativo Fedora (Linux). El tiempo obtenido para encontrar la solución exacta fue de 79 segundos.

El algoritmo paralelo requiere de la determinación del número de iteraciones necesario para revisar la solución optima. Experimentalmente determinamos que ese número es de 100 para un procesador y de 5 cuando utilizamos los 31 procesadores. El motivo es, cuando utilizamos un solo procesador los cubos se almacenan en una lista ligada hasta que les toca el momento de ser calculados, similar al algoritmo de la versión secuencial. En cambio, cuando se tienen los 31 procesadores se realiza la misma búsqueda de la solución optima, pero con la diferencia de que cada procesador lo hace en los cubos que le tocan y que los pone en su lista correspondiente. Con estas condiciones ambas versiones obtienen la misma solución exacta.

Para determinar si el algoritmo paralelo es rápido y eficientemente paralelizado, definimos [13]:

- Aceleración paralela: $S_p = \frac{T_1}{T_p}$

- Eficiencia paralela: $E_p = \frac{S_p}{p}$

Para el problema de *m = 500,* el tiempo obtenido en un procesador de la IBM fue de *0.6317* segundos, esto es *125* veces mas rápido que el obtenido en un procesador de una computadora personal. Cuando utilizamos los *31* procesadores el tiempo de ejecución es de *0.488* segundos, esto es $S_p = 1.29$ y una eficiencia de $E_p = 4.16\%$, lo cual nos indicaría que el algoritmo paralelo es malo. Esto cambia cuando incrementamos el número de variables a *m = 1000* y determinamos la solución optima para la misma λ. El tiempo de computo obtenido en un procesador es de *3.2426* segundos y el tiempo utilizando *15* procesadores es de *0.5954* segundos, esto nos da $S_p = 5.45$ y una eficiencia paralela $E_p = 45.42\%$.

Los resultados anteriores muestran que el tiempo utilizado para sincronizar el envío de mensajes es significativo, ya que el tamaño de los mensajes no es grande, como se describió previamente. Para el problema de la selección optimal del subconjunto de objetos, cuando tenemos una *m* no muy grande, el tiempo utilizado para el cómputo no es significativo. Cuando *m* crece se muestra que el tiempo de cómputo se incrementa y la computadora paralela es mas eficiente. Lo cual nos indica que para problemas de otro tipo con mayor número de restricciones que requieran de mas computo el algoritmo paralelo será mas eficiente. Esto nos alienta ya que la finalidad del método del árbol de cubos, sus algoritmos secuencial y paralelo es determinar la solución optima de problemas de grandes dimensiones.

6. Conclusiones

El método y los algoritmos, secuencial y paralelo, del árbol de cubos son eficaces para resolver problemas de optimización discreta. En la gran cantidad de problemas resueltos se determino la solución exacta sin requerir un número elevado de iteraciones, en el peor de los casos fue $O(m^4)$.

El método del árbol de cubos, fundamentado matemáticamente, es una buena opción para resolver problemas que se venían resolviendo con otros métodos exactos, pero con la diferencia de que nuestro método se puede aplicar a solucionar problemas de optimización discreta de dimensiones mayores. Además, en el caso de que el problema sea lo suficientemente grande y requiera de una gran cantidad de memoria y procesamiento, que no sea posible satisfacerla con una computadora de un solo procesador, es posible utilizar el método con una variante para determinar la solución aproximada, donde la solución optima estará en el intervalo dado por la precisión requerida.

En caso de requerir la solución exacta es posible utilizar al algoritmo paralelo del árbol de cubos, donde nuestros primeros resultados nos indican que entre mas cómputo sea necesario mas eficiente es el algoritmo paralelo.

Finalmente, con el problema de la selección optimal del subconjunto de objetos se muestra que los problemas cotidianos en una empresa pueden ser muy complejos y no computables si no se tiene la tecnología, métodos y algoritmos adecuados para resolverlos.