Rev. Téc. Ing. Univ. Zulia. Vol. 21, Nº 1, 47-58, 1998

Parallel genetic algorithms on combinatorial optimization problems

Francisco Hidrobo and José Aguilar*

Laboratorio SUMA, Facultad de Ciencias, Fax: 074-401295, email: hidrobo@ciens.ula.ve.
**CEMISID. Facultad de Ingeniería, Fax: 074-401295, email:aguilar@ing.ula.ve.*
Universidad de los Andes. La Hechicera, 5101. Mérida, Venezuela

Abstract

In this work, we introduce several parallel approaches based on Genetic Algorithms to solve NP-complete problems. Each approach uses a different concept of the parallel computing. A first approach exploits the implicit parallelism in the internal operation of this technique. The second approach perform a decomposition of the solutions space in order to carry out a detailed search through each one of them using several Genetic Algorithms. The last approach proposes a reinforced algorithm of search for the Genetic Algorithms based on concepts of the Collective Intelligence theory. In order to prove and compare the parallel approaches, the Graph Partitioning and Travelling Salesman Problems are studied. The parallel library used is PVM (Parallel Virtual Machine) and the tests were performed on a SP2-IBM with 8 nodes.

Key words: Genetic algorithms, parallel computing, combinatorial optimization Problems.

Algoritmos genéticos paralelos en problemas de optimización combinatoria

Resumen

En este artículo presentamos varias heurísticas paralelas basadas en Algoritmos Genéticos, las cuales resuelven problemas de Optimización Combinatoria. Cada una de las heurísticas explotan diferentes conceptos de paradigma de programación paralela, tales como descomposición del espacio de datos, explotación del paralelismo implícito o uso de conceptos ligados a la inteligencia colectiva (distribuida). Estos algoritmos fueron desarrollados usando PVM (Parallel Virtual Machine) como ambiente de programación paralelo, probados en los problemas combinatorios Partición de Grafos y Viajero de Comercio, y ejecutados en una SP2 de IBM con 8 nodos.

Palabras clave: Algoritmos genéticos, computación paralela, optimización combinatoria.

Introduction

La utilización de los computadores para la solución de problemas complejos ha dado cabida a numerosos estudios, fundamentalmente en cuanto a mecanismos (métodos, algoritmos, estrategias, etc.) de solución se refiere. Un ejemplo de esto es el uso de "técnicas" inspiradas en los Sistemas Biológicos como base para el desarrollo de heurísticas que resuelven problemas NP-complejos. Entre las técnicas más utilizadas destacan: los Algoritmos Genéticos [1-5] y las Redes Neurales [1].

La solución de problemas NP-complejos usando cualquier tipo de programa de resolución normalmente consumen una gran cantidad de recursos de computación. En el caso de los métodos analíticos, si bien estos permiten siempre conseguir la solución exacta, se hacen prohibitivos en cuanto a tiempos de ejecución cuando el

tamaño del problema es grande [1, 6]. En el caso de los métodos aproximativos o heurísticos, estos logran obtener buenos resultados (quizás la solución exacta) en tiempos de ejecución razonables. El dilema al que uno se enfrenta con estos métodos es en la escogencia entre una buena calidad de las soluciones con un tiempo de ejecución largo o calidades de soluciones "regulares" y tiempos de ejecuciones cortos.

Sin embargo, los Ambientes de Procesamiento Paralelo han abierto la posibilidad de mejorar los métodos heurísticos (a nivel de calidades y/o tiempos de ejecución) e incluso han hecho posibles implementaciones de heurísticas de resolución no "posibles" en plataformas secuenciales por los tiempos de ejecución que estas implicaban. Explotar el paralelismo implícito de estos métodos, así como utilizar conceptos y modelos antes no posibles por el tipo de procesamiento que se derivaba, es ahora posible en estas plataformas. Los Algoritmos Genéticos han sido usados con éxito como métodos de búsqueda y optimización [7, 8]. Los Algoritmos Genéticos han sido paralelizados según tres enfoques:

a. Siguiendo el modelo estándar, pero efectuando las fases de evaluación y reproducción en paralelo [9].

b. Un modelo de descomposición, el cual consiste en dividir la población en subpoblados y ejecutar el algoritmo estándar sobre cada subpoblación [10].

c. Un modelo de reproducción local que consiste en colocar cada individuo sobre un procesador y la reproducción se hace con sus vecinos [11]; el grado de paralelismo de este esquema es más fino que el de los anteriores.

Dentro de los enfoques de paralelización se han destacado las implementaciones asíncronas [11], de manera de no interrumpir los cómputos esperando la comunicación con algún otro procesador. Dentro de esta área se destacan los trabajos realizados por Barán et al. [12], los cuales combinan algoritmos en un ambiente asíncrono usando una técnica conocida como *A-Teams (Asynchronous Teams)*.

En este trabajo se presentan varios esquemas de Heurísticas basadas en Algoritmos Genéticos que resuelven problemas NP-Complejos. Un primer esquema explota el paralelismo implícito en las operaciones internas de esta técnica. El segundo esquema realiza una descomposición del espacio de soluciones para hacer una búsqueda más detallada a través de cada uno de ellos, usando para ello varios Algoritmos Genéticos. El último esquema propone un algoritmo reforzado de búsqueda para el Algoritmo Genético basado en conceptos de Inteligencia Colectiva. En este caso, lo que se busca es crear una memoria colectiva de los mejores cambios estructurales en los individuos a través de las generaciones, para usar esa información como rastro de búsqueda. El trabajo tiene como principal aporte el hacer una evaluación de tres enfoques distintos de paralelización de los Algoritmos Genéticos, utilizando como punto, principal, de evaluación la calidad de la solución.

Para probar y comparar los esquemas paralelos, se estudian los problemas de Partición de Grafos y Viajero de Comercio. La herramienta de paralelización usada es PVM (Parallel Virtual Machine) y las pruebas fueron realizadas sobre una SP2 con 8 nodos.

El resto del trabajo es organizado como sigue. En la sección 1 se hace una breve descripción de los Algoritmos Genéticos. Enseguida presentamos los esquemas de paralelización de los Algoritmos Genéticos. La sección 3 presenta el problema de Partición de Grafos, las comparaciones y el análisis de resultados entre los diferentes enfoques paralelos. La sección 4 presenta el problema del Viajero de Comercio, las comparaciones y el análisis de resultados entre los diferentes enfoques paralelos. Finalmente, presentamos las conclusiones y futuros trabajos.

Algoritmos Genéticos (AG)

Los AG fueron propuestos por John Holland a comienzos de los años 70 [1, 7, 8]. Son una técnica de búsqueda adaptativa que ha sido intensamente estudiada estos últimos años. Emulan la evolución biológica en el computador siguiendo un proceso *evolutivo inteligente* sobre individuos, basado en la utilización de operadores evolutivos tales como mutación, cruzamiento, selección e inversión.

El procedimiento general de todo AG es el siguiente: un AG mantiene una población de po-

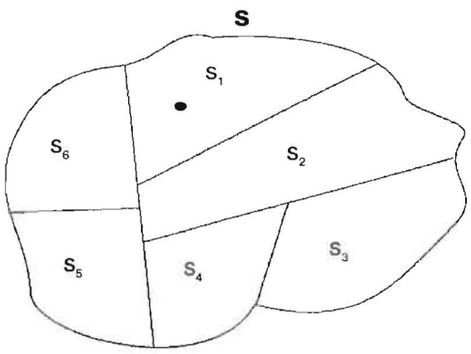

Figura 1. Particionamiento del Espacio de Soluciones.

tenciales soluciones a través de las generaciones. Cada solución es evaluada usando una función objetivo para determinar que tan buena es. Entonces, una nueva población es construida seleccionando las mejores soluciones de la población actual, las cuales son usadas para ser reproducidas por medio de la aplicación de los operadores evolutivos y generar nuevas soluciones. La idea de fondo es encontrar el mejor óptimo global, partiendo de un conjunto de soluciones escogidas aleatoriamente. Para eso se utilizan los operadores evolutivos como mecanismos de búsqueda de nuevas quizás mejores, soluciones. El procedimiento continua hasta ser atrapado en un óptimo local. El macroalgoritmo es el siguiente:

1. Generar una población inicial (conjunto de soluciones iniciales para el problema).
2. Evaluar las soluciones (cada individuo).
3. Seleccionar algunos individuos de la población (soluciones) para generar nuevas soluciones aplicando los operadores genéticos.
4. Reproducir y evaluar estas nuevas soluciones.
5. Reemplazar los peores individuos de la población por los mejores individuos producto de la reproducción.
6. Verificar el criterio de convergencia, o regresar al paso 3.

Esquemas Paralelos

En esta sección presentamos los tres enfoques de paralelización que utilizaremos.

Descomposición del espacio de soluciones

Este enfoque consiste en dividir el espacio de soluciones en varios subespacios y en ejecutar un AG en cada uno de ellos. Bajo este enfoque se tendrán tantos AG en ejecución como subespacios de soluciones se tengan. Con este enfoque, el objetivo final es paralelizar la búsqueda para hacerla más exhaustiva. De esta manera es posible encontrar varias soluciones subóptimas, pudiendo ser una de ellas óptima. Así, este enfoque evita el problema clásico de los AG secuenciales en cuanto al alto grado de dependencia de las soluciones iniciales utilizadas (población inicial).

Suponiendo que S es el espacio de soluciones, la manera como se realiza el particionamiento (Figura 1) de S en K subespacios $S_i \; \forall i = 1, \ldots, K$ cumple con las siguientes condiciones:

— $S_i \subset S \forall i = 1, \ldots, K$.

— $S_i \cap S_j = \varnothing \forall i, j = 1, \ldots, K$ de lo contrario $i = j$

— $S = \bigcup_1^K S_i$

Explotación del paralelismo implícito de los AG

En este caso, lo que se busca es paralelizar la estructura interna de los AG para ejecutar simultáneamente todas las operaciones que se puedan [6]. Las etapas básicas de los AG vienen dadas por los pasos principales del macro-algoritmo: generación de la población inicial, Evaluación, Selección, Reproducción y Reemplazo. La paralelización se basa en la independencia observada en cada una de las etapas básicas. Estamos entendiendo por paralelismo implícito el hecho de descomponer el algoritmo genético en procesos/tareas las cuales pueden ser paralelizadas ya que sus estructuras internas son intrínsecamente paralelas (por ejemplo, se puede hacer simultáneamente la evaluación de los individuos), este tipo de paralelismo difiere de la noción de paralelismo implícito que propone Holland en su trabajo [Golberg89]. Así, los siguientes procesos se pueden paralelizar:

— La generación inicial de individuos,

— La evaluación de cada individuo,

— La reproducción de los individuos (creando de manera concurrentes los nuevos individuos).

Partiendo de las etapas que se pueden paralelizar, el algoritmo general paralelizado es de la siguiente forma:

1. General *paralelamente* los individuos de la población inicial (soluciones iniciales al problema).
2. Evaluar *paralelamente* las soluciones (individuo)
3. Seleccionar *secuencialmente* los L mejores individuos.
4. Reproducir *paralelamente* los nuevos individuos usando los individuos escogidos.
5. Evaluar *paralelamente* las nuevas soluciones.
6. Reemplazo *secuencialmente* los individuos.
7. Verificar la Convergencia del algoritmo.

Algoritmo reforzado de búsqueda

En este esquema se presenta un algoritmo reforzado de búsqueda basado en la inteligencia colectiva [2-4]. La idea de base es la siguiente: se descompone el espacio de las soluciones de la forma expuesta en el enfoque basado en descomposición del espacio de soluciones y se utiliza una seria de reglas para definir la información a transmitir entre los subespacios de soluciones. La información a transmitir esta conformada por los detalles estructurales encontrados en los mejores individuos durante la búsqueda en cada uno de los subespacios. Es para la realización de las reglas en que nos hemos inspirado en la Inteligencia Colectiva, particularmente en el caso de las Colonias de Hormigas.

A nivel general, el comportamiento de las Colonias de Hormigas para cumplir sus objetivos de supervivencia es "impresionante". Esto no puede atribuirse a las capacidades cognoscitivas de cada hormiga, sino derivado de un proceso **Colectivo Inteligente**. Dicho proceso está basado en las capacidades comunicacionales de las hormigas, que definen las interrelaciones entre ellas. Estas interrelaciones permiten transmitir las informaciones de interés que va encontrando cada Hormiga, de tal manera de cumplir con los objetivos de la Colonia. Esta información es transmitida en forma de trazas, denominadas **pheromone.** Por ejemplo, para cumplir el objetivo de alimentarse, las hormigas inician un proceso aleatorio de búsqueda de fuentes de alimentos. En medio de esta búsqueda, ellas van dejando trazas que van indicando los caminos que han recorrido. Después de que una Hormiga encuentra una fuente de alimentos, ella regresa al nido. El camino que ha dejado marcado con su traza, podrá ser tomado por ella y otras hormigas para explotar la fuente de alimentos encontrada, convirtiéndose en el tiempo en un proceso de búsqueda determinística de alimentos [2-4].

T_{11}	T_{12}	T_{13}	T_{14}	T_{15}	T_{16}	.	.	.	T_{1n}
T_{21}	T_{22}	T_{23}	T_{24}	T_{25}	T_{26}	.	.	.	T_{2n}
T_{p1}	T_{p2}	T_{p3}	T_{p4}	T_{p5}	T_{p6}	.	.	.	T_{pn}

Figura 2. Individuo Traza.

Usando estas ideas, nosotros distribuiremos la búsqueda de subespacios denominados **agentes y hormigas.** En nuestro caso, cada subespacio ejecuta un AG. Además, seguimos el comportamiento estructural de los mejores individuos de las poblaciones durante sus evoluciones para cada AG (es decir, en cada subespacio), a través de lo que hemos denominado **individuos trazas.** Así los individuos trazas llevan el registro de los buenos cambios convirtiéndose en la "memoria colectiva" del sistema.

Esta es la gran diferencia con respecto a los AG clásicos. En los AG clásicos no se lleva un registro del proceso evolutivo de los individuos, es decir, una vez que en una generación se ha cambiado la información de la población actual, esta no se toma en cuenta en futuras generaciones. Lo que se propone es llevar en cada subespacio de soluciones un **individuo traza,** que guarde la información pasada, definido según la Figura 2.

Las columnas del individuo traza representan los elementos de cada individuo, mientras que las filas son los distintos valores que estos elementos pueden tomar. Cada elemento T_{jl} tendrá un valor definido por el siguiente procedimiento:

— Se toma el individuo i en el tiempo k con valor de su función objetivo de $F_i(k)$ en el tiem-

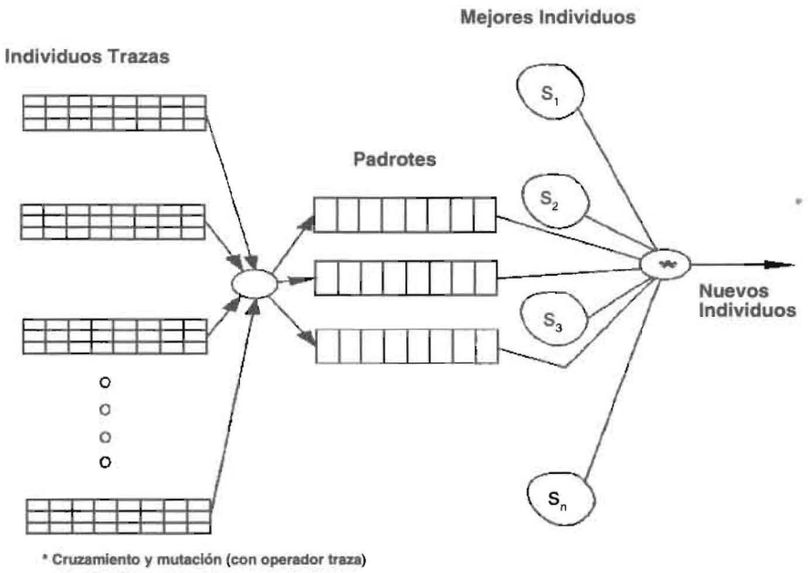

Figura 3. Mezcla de los resultados parciales.

po $k+1$ con valor de su función objetivo de $F_i(k+1)$.

— Se define un valor de mejora por cambios, expresado de la siguiente manera:

$$F_t = \frac{(F_i(k+1) - F_i(k))}{c}$$

donde c es el número total de elementos que cambiaron en el individuo i.

Como lo que se desea tomar es la cantidad en la cual el individuo i ha mejorado, dependiendo si el objetivo es minimizar o maximizar, se considera interesante el cambio si F_t es positivo y el objetivo es maximizar, o viceversa si el objetivo es minimizar.

— El valor de T_{jl} se incrementará en F_t si el elemento l del individuo i ha cambiado al valor j.

El individuo traza registrará a través de la función matemática (F_t) el aporte de los buenos individuos (incluso de cada uno de sus elementos) reproducidos con respecto a los individuos que estos reemplazan en la población original.

El concepto fundamental que se usa en la definición de la función matemática mencionada anteriormente es el mismo que se presenta en los sistemas de hormigas: **pheromone**. Dicha función representa el valor de mejora en la solución buscada a través de las generaciones. La idea principal es reflejar en el individuo traza las mejoras que cada nuevo individuo presenta, tanto en el aspecto global como en sus componentes específicos, es decir, reflejar que parte del individuo es la que hace que sea mejor con respecto a los individuos reemplazados.

Reglas de transmisión de la información

Uno de los aspectos más resaltantes es como utilizar la información guardada en los individuos trazas para guiar el proceso de búsqueda por los caminos que cada subespacio ha ido determinando como interesantes. En menos palabras, como se deben mezclar los resultados obtenidos en cada subespacio a partir de los patrones generados en cada uno de ellos (Figura 3).

El objetivo de esta mezcla es tratar de usar la información recabada por las trazas como un elemento que guíe el procedimiento de búsqueda hacia regiones mejores que las anteriores. De esta manera, se proponen dos mecanismos o reglas (no excluyentes):

— Generación de individuos **padrotes** a partir de la información de los individuos trazas. Los padrotes están compuestos por los me-

jores valores identificados en los individuos trazas. Dicha identificación usará un esquema de probabilidad según los valores más altos que contengan los individuos trazas. Entonces, se usan estos padrotes para aplicarles los operadores genéticos que se estén usando en el problema bajo estudio. Así, cada aplicación de un operador genético podrá seleccionar los individuos sobre los cuales trabajará de entre los padrotes y los mejores individuos de cada sub-espacio.

— Definición de un nuevo operador evolutivo, denominado **operador traza**, el cual es una modificación de alguno de los operadores genéticos que se están usando en el problema bajo estudio. El operador que se escoja para modificar, debe ser alguno que permita realizar saltos en el espacio de búsqueda (por ejemplo, mutación o inversión),, pero en este caso ese salto será dirigido de manera de cambiar los valores de los elementos que componen al individuo en evolución, hacia los mejores valores identificados en el individuo traza. Los mejores valores son aquellos que tuvieron mejor comportamiento (mayor valor de T_{jl}).

Después se deben discretizar los nuevos individuos generados en las particiones del espacio de soluciones (Figura 1). Por lo tanto, serán migrados solo al subespacio que les corresponda.

Partición de Grafos

Este problema consiste en dividir un grafo G en varios subgrafos, tal que se minimice una función objetivo [1, 6]. Una posible definición es la siguiente: dado un grafo G, lo que se desea es partir sus nodos en K subgrafos de tal manera que el número de nodos por subgrafo sea igual y el número de arcos conformados por nodos que estén en diferentes subgrafos sea mínimo. Matemáticamente, el problema de partición de grafo puede ser definido como:

$G = (N, A,)$,

donde: $N = \{1, ..., n\}$ es el grafo con n nodos; $A = \{a_{ij}\}$ es la matriz de adyacencia,

La función objetivo asocia un valor para cada posible partición. Así, nosotros proponemos la siguiente función:

$$F_c = \sum_{i,j \in D} a_{ij} + b \frac{\sum_{z=1}^{K} (N_{G_z} - n/K)^2}{K}$$

donde:

$D = \{i \in G_m \ \& \ j \in G_l \ \& \ l \neq m \ \& \ a_{ij} = 1\}$

N_{G_z} = número de nodos en subgrafo z

b = factor de equilibrio [0,2].

El primer término representa el costo de comunicación o corte entre subgrafos y el segundo el costo de desequilibrio de carga. El problema consiste en conseguir la partición del grafo que minimice el valor de la función de cost F_c.

Resolución usando los algoritmos genéticos

El enfoque usado para resolver este problema usando Algoritmos Genéticos es basado en los trabajos [1, 6, 13]. Para el problema de partición de un grafo de N nodos en K particiones se utiliza la matriz de adyacencia como representación del grafo. Para codificar los individuos se definen estos como un vector de longitud N, donde cada celda del vector representa un nodo del grafo y el valor que ella almacena corresponde a la partición (subgrafo) a la cual se asigna el nodo en esa solución. Por ejemplo, si se supone N=5 y K=2 y se tiene un individuo con los siguientes valores: 2,1,1,2,1; eso significa que en esa solución el nodo 1 está en el subgrafo 2, el nodo 2 en el subgrafo 1, el nodo 3 en el subgrafo 1 y así sucesivamente [1]. La función objetivo que se utiliza es la que se viene a definir (ecuación 2). Los detalles de los operadores genéticos utilizados son los siguientes:

— Cruzamiento. Este operador permite combinar dos individuos. Se toma aleatoriamente una posición (j) como el punto que corta los dos individuos. Luego, se intercalan las dos mitades de los individuos para generar dos nuevos individuos.

— Mutación. Con este operador se cambia aleatoriamente un valor (o varios) del individuo. Se escoge aleatoriamente la parte a

cambiar y se generan (mutan) aleatoriamente los nuevos valores de esa parte.

Las probabilidades de uso de cada operador se obtuvieron de manera experimental para cada par N, K (con el programa serial). Tomando los valores que proporcionaban una mejor solución.

Los detalles más relevantes de las implementaciones paralelas para este problema son los siguientes:

— La selección del esquema de particionamiento que se hace para este caso toma una posición del arreglo (por ejemplo, tercer nodo) de manera aleatoria y asigna a cada subespacio un valor diferente del subgrafo donde podrá ser asignado ese nodo (entre 1 y K).

— Para el caso del algoritmo de reforzamiento:

- La información que refleja el *individuo traza* contiene los mejores valores (entre 1 y K) donde deberían ser asignados cada uno de los nodos.

- *El operador traza* es una modificación del operador de mutación. Este operador permitirá realizar una mutación dirigida al cambiar los valores de los elementos que componen al individuo hacia los mejores valores identificados en el individuo traza. Los mejores valores son aquellos que tuvieron mejor comportamiento (mayor valor de T_{jl}). Así, se escogen aleatoriamente varios elementos de los individuos y se mutan sus valores a los mejores valores encontrados para esos elementos.

- Los individuos *padrotes* que se generan se usan para emparejarlos con los mejores individuos de cada subpoblación (es decir, de cada subespacio) en las operaciones de cruzamiento. Así, en cada operación de cruzamiento es escoge aleatoriamente a un padrote para cruzarse con alguno de los individuos de alguna de las subpoblaciones.

Análisis de resultados

La implementación se realizó usando el lenguaje C y la biblioteca PVM bajo un esquema maestro-esclavo, donde el maestro realiza las tareas de iniciación y creación de los procesos específicos a ejecutar simultáneamente (e.j. la evaluación de individuos en la versión de paralelismo implícito).

La plataforma de ejecución fue la máquina IBM-SP2 del Centro de Cálculo Científico de la ULA, que posee 8 procesadores. Los resultados son representativos del problema bajo estudio. En este caso, cada punto de las gráficas representa el promedio de 30 simulaciones para los diferentes conjuntos de valores de n y K. Los resultados se presentan en la Tabla 1.

Puede observarse de los resultados que los tiempos de ejecución de los programas paralelos (AGPI, AGP y AGPR) se incrementan derivados de la herramienta utilizada como plataforma de programación la cual se fundamenta en el esquema de pase de mensajes. Esto introduce un nuevo tiempo, conocido como tiempo de comunicación (como era de esperarse) que puede ser causado por: número de comunicaciones de los procesos, problemas de latencia de PVM. Además, el programa AGRP siempre tarde más que AGP, este tiempo adicional se produce por la manipulación que debe hacer AGPR de los individuos trazas (Reglas de transmisión de la información). Estableciendo una comparación entre los resultados de AGPI y AGS se observa una gran similitud en los mismos, determinándose que efectivamente AGPI no mejora la calidad de los resultados. Por otro lado, AGPI introduce un incremento significativo en el tiempo de ejecución, lo que reduce el interés en este esquema de paralelización. Dicho incremento es debido a la pequeña granularidad de los procesos que componen el programa paralelo, y al alto nivel de comunicaciones entre esos procesos y el programa maestro.

El punto más resaltante es la calidad de los resultados obtenidos, en los cuales se ha logrado mejoras cualitativas. Así, en cuanto a la calidad de las soluciones, tanto AGP como AGPR encuentran mejores soluciones que AGS. La Tabla 8 muestra que AGPR encuentra una mejor solución que AGP, en las otras tablas estas mejoras son más pequeñas. La contribución de las implementaciones paralelas en la calidad de las soluciones viene derivada de poder hacer una búsqueda más exhaustiva y rigurosa.

Tabla 1
Resultados para el problema de partición de grafos

a.n=20 K=4

ALG	MIN	MAX	PROM	t(seg)
AGS	50	54	52	0.150
AGPI	49	53	51.8	5.127
AGP	46	50	48.8	0.440
AGPR	46	51	48.5	3.900

b. n=30 K=5

ALG	MIN	MAX	PROM	t(seg)
AGS	129	136	131	0.400
AGPI	130	135	131.2	8.737
AGP	124	126	124.6	1.294
AGPR	124	127	125.8	9.830

c. n=40 K=6

ALG	MIN	MAX	PROM	t(seg)
AGS	28	29	28.4	1.540
AGPI	27	29	27.8	15.307
AGP	26	28	26.6	4.760
AGPR	26	28	26.8	25.200

d. n=50 K=7

ALG	MIN	MAX	PROM	t(seg)
AGS	271	278	272.2	3.840
AGPI	270	278	274.3	25.228
AGP	263	268	266	11.760
AGPR	262	266	264.4	50.910

e. n=60 K=8

ALG	MIN	MAX	PROM	t(seg)
AGS	479	489	484.8	8.530
AGPI	480	493	486.8	40.972
AGP	470	481	477.6	22.610
AGPR	473	479	476.4	98.220

f. n=70 K=9

ALG	MIN	MAX	PROM	t(seg)
AGS	756	765	760.8	14.300
AGPI	754	772	767.8	54.282
AGP	731	741	737	37.680
AGPR	726	745	740.5	160.750

g. K=5

ALG	N=20	N=50	N=100	N=200
AGS	59	250	1540	1787
AGPI	54	249	1542	1786
AGP	51	246	1520	1772
AGPR	51	244	1517	1772

h. n=100

ALG	K=3	K=4	K=5	K=6	k=7
AGS	1268	1430	1535	1601	1638
AGPI	1270	1430	1536	1603	1636
AGP	1267	1426	1525	1586	1624
AGPR	1265	1416	1520	1580	1616

AGS=Implementación Serial. AGPI=Implementación Paralela usando el paralelización Interna. AGP=Implementación Paralela usando el esquema de particiones. AGPR=Implementación Paralela usando el esquema de reforzamiento del AG. n=Número de Nodos del Grafo. K=Número de particiones.

Agente Viajero

El problema del agente viajero es un problema clásico de optimización combinatoria que puede describirse como: *Dadas n ciudades, el viajero de comercio debe visitar cada ciudad una vez y sólo una vez, teniendo en cuenta que el costo total del recorrido debe ser mínimo.*

Una de las instancias más comunes del problema del agente viajero es el problema *euclidiano*, donde el costo del recorrido viene dado por la suma ordenada de las distancias entre las ciudades visitadas. Este problema puede ser expresado de la siguiente manera:

$G = (N, A)$

donde: $N = \{1, ..., n\}$ es el grafo con n nodos; $A = \{a_{ij}\}$ es la matriz de adyacencia.

Además, se define la matriz de distancias D:

$$\{d_{ij}\} = \begin{cases} \infty & \text{Si } a_{ij} = 0 \\ l_{ij} & \text{Si } a_{ij} = 1 \end{cases}$$

donde: l_{ij} = distancia entre las ciudades i y j.

Suponiendo que las ciudades son numeradas desde 1 hasta N, una solución al problema puede expresarse a través de una matriz de estado (E) que indique el orden en que son visitadas las ciudades. Esta matriz tendrá en las filas el orden de visita de las ciudades y en las columnas las ciudades.

$$\{e_{ij}\} = \begin{cases} 1 & \text{Si la ciudad } j \text{ fue la } i\text{-esima ciudad visitada} \\ 0 & \text{En otro caso} \end{cases}$$

La matriz de estado E permitirá verificar la validez de una solución, de tal manera de asegurar que todas las ciudades son visitadas una y sólo una vez. Esto se hace por la verificación de las siguientes restricciones:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} e_{ij} = n \qquad (3)$$

$$\sum_{i=1}^{n} e_{ij} = 1 \qquad (4)$$

$$\sum_{j=1}^{n} e_{ij} = 1 \qquad (5)$$

Así mismo, la matriz E permite definir un arreglo unidimensional V de dimensión "n", dicho arreglo contendrá en cada posición la ciudad que fue visitada en dicha posición.

$v_j = i$ (Si la ciudad j fue la i-esima ciudad visitada).

Por último, para evaluar cada posible solución se propone una función objetivo compuesta por dos partes, una corresponde a los costos relativos a las distancia recorridas y el otro relativo al grado de validez de la solución. Tales funciones serían:

$$F_1 = \sum_{i=1}^{n} \sum_{k=1}^{n} \sum_{j=1}^{n} l_{ik} e_{ij} e_{kj+1} \qquad (6)$$

$$F_2 = C\left(\left| \sum_{i=1}^{n} \sum_{k=1}^{n} e_{ik} - n \right| + \sum_{i=1}^{n} \left| \sum_{k=1}^{n} e_{ik} - 1 \right| + \sum_{k=1}^{n} \left| \sum_{i=1}^{n} e_{ik} - 1 \right| \right) \qquad (7)$$

$$F_C = F_1 + F_2 \qquad (8)$$

donde: C = Max(l_{ik}*n) (factor de penalización).

Resolución usando los algoritmos genéticos

El problema consiste en encontrar el recorrido de las ciudades que minimice el valor de la función de costo F_C. Con el planteamiento de esta función objetivo es posible proponer dos alternativas para la implementación de un AG que resuelva el problema del viajero de comercio:

1. Utilizar "todos" los operadores genéticos, realizándose la verificación de la validez de cada solución a través del valor correspondiente de la función objetivo. En esta alternativa se usaría la función $F_C = F_1 + F_2$.
2. Hacer uso de los operadores de inversión y traslocación como únicos operadores genéticos. De esta manera, todas las soluciones generadas a partir de soluciones válidas serán válidas. Así, la función F_2 no será necesaria quedando que $F_C = F_1$. En este caso, la validez de las soluciones iniciales debe ser verificada en el propio proceso de generación.

La forma de representación de los individuos (soluciones) se realiza por medio de un arreglo unidimensional de dimensión n (número de ciudades), el cual representa el arreglo V definido previamente. Así, dicho arreglo contiene el recorrido ordenado, es decir, en la posición 1 irá la primera ciudad visitada, en la posición 2 la segunda, y así sucesivamente. Para utilizar esta representación se supone que las ciudades son numeradas de 1 a n.

Nosotros implementamos la estrategia dos puesto que es más sencilla de implementar, así la función F_1 será la que utilizaremos en el algoritmo genético. La verificación de las restricciones para el problema son realizadas a través de los operadores genéticos utilizados. Así, los operadores cumplen dos funciones: la de evolución genética y la de generar soluciones válidas. Los detalles de los operadores utilizados son los siguientes:

— Inversión. En este caso se toma aleatoriamente una posición (j) (debe tener un valor de ciudad c_1) y se genera un nuevo valor para la ciudad (c_2) que es visitada en esa posición, también de forma aleatoria. Luego, se busca este nuevo valor de la ciudad en el

arreglo (debe ocupar una posición i) y a la posición se le asigna el valor c_1.

— Traslocación. Con este operador lo que se hace es un desplazamiento. Se supone que el arreglo que representa la solución es circular y lo que se hace es correr los valores de las posiciones en que son visitadas las ciudades.

Las propuestas e implementaciones para el problema del agente viajero son esencialmente las mismas, las posibilidades de los operadores se obtuvieron con el mismo mecanismo que para el caso de partición de grafos.

Los detalles más relevantes de las implementaciones paralelas para este problema se presentarán a continuación:

— La selección del esquema de particionamiento que se hace para el caso del agente viajero toma una posición de visita de manera aleatoria (por ejemplo, cuarta ciudad a ser visitada) y asigna a cada subespacio un valor diferente de ciudad que se visitará en esa posición.

— Para el caso del algoritmo de reforzamiento:

- La información que refleja el *individuo traza* contiene la mejora que representa la visita de las ciudades en determinadas posiciones.

- En el agente viajero el *operador traza* es una modificación del operador de inversión. Este operador permitirá realizar una inversión dirigida al seleccionar aleatoriamente una posición y cambiar su valor hacia los mejores valores (mayor valor de T_{jl}) identificados en el individuo traza. Esto conlleva a modificar a posteriori a la posición que tenía ese valor y asignarle el valor que tiene la posición que se escogió aleatoriamente.

- Los individuos *padrotes* que se generan son seleccionados aleatoriamente para aplicarles los operadores de inversión o traslocación. Además, grupos de padrotes son introducidos en cada uno de los subespacios aleatoriamente (según, si pertenecen a ese subespacio o no).

Análisis de resultados

El procedimiento experimental para el estudio del problema del viajero de comercio es el mismo que se utilizó para el caso de partición de grafos. Sin embargo, para el viajero de comercio no se implementó el programa con paralelización de la estructura interna (AGPI)puesto que los resultados obtenidos en el problema de partición de grafos mostraron que este algoritmo incrementa el tiempo sin mejorar la calidad de la solución.

Los resultados obtenidos para el caso del Agente Viajero son muy semejantes a los que se obtuvieron para el problema de Partición de Grafos. En todas las pruebas los resultados de las implementaciones paralelas (AGP y AGPR), son mejores que el de la implementación serial (AGS). Para todos los casos, a excepción de los presentados en la Tabla 2.b, el algoritmo reforzado paralelo (AGPR) encuentra una mejor solución que la versión paralela (AGP).

Por otro lado, en lo referente al tiempo de ejecución de los algoritmos genéticos, se observa que AGPR siempre es más lento que AGP; sin embargo, AGP no siempre es más lento que AGS. De las Tablas 2.e y 2.f se puede observar que el tiempo de ejecución de AGP es menor al de AGS. Esta mejora de tiempo en la implementación paralela puede atribuirse al tamaño del problema. La mejora en tiempo en cuanto al tamaño del problema se debe al aumento en la granularidad de los procesos paralelos involucrados.

Conclusiones

La utilización de una biblioteca de pase de mensajes (PVM) para la paralelización de los AG que resuelven los problemas de optimización no mejora el tiempo de ejecución. Es probable que la reducción en el tiempo de ejecución se haga presente en ambientes paralelos a memoria compartida o utilizando una biblioteca de pase de mensajes más eficiente. Además, nuestros programas necesitan sincronización a nivel del maestro por la espera de resultados en cada iteración, lo que implica que, eventualmente, se pueden obtener resultados en tiempos cortos sin poder ser explotados. Próximas versiones deberán corregir esto a través de un enfoque asincrónico.

La contribución de las implementaciones paralelas se hace visible en la calidad de la solu-

Tabla 2
Resultados para el problema del agente viajero

a. n=10

ALG	MIN	MAX	PROM	t(seg)
AGS	43	66	52.933	0.032
APG	38	58	45.033	0.994
AGPR	33	49	42.700	1.986

b. n=20

ALG	MIN	MAX	PROM	t(seg)
AGS	135	217	174.867	0.120
AGP	130	180	152.500	0.903
AGPR	123	164	147.567	4.382

c. n=50

ALG	MIN	MAX	PROM	t(seg)
AGS	509	524	516.867	0.503
AGP	448	512	487.433	4.132
AGPR	461	501	483.167	14.715

d. n=100

ALG	MIN	MAX	PROM	t(seg)
AGS	603	688	645.200	92.239
AGP	586	640	613.150	108.971
AGPR	583	655	614.400	435.139

e. n=200

ALG	MIN	MAX	PROM	t(seg)
AGS	1627	1992	1892.500	469.545
AGP	1539	1931	1654.600	322.120
AGPR	1528	1662	1611.400	1702.155

f. n=500

ALG	MIN	MAX	PROM	t(seg)
AGS	5184	5184	5184	1408.570
AGP	4818	4962	4912.700	1108.549
AGPR	4781	5184	4872.909	6093.642

ción, ya que las mismas permiten realizar una búsqueda más rigurosa y exhaustiva del espacio global.

En particular, el esquema de reforzamiento planteado efectivamente permite hacer un seguimiento de la evolución de los individuos en los AG, para a posteriori explotar dicha información en la fase de reproducción a través de diferentes mecanismos (generación de padrotes y operador traza), lo que conduce a mejorar la calidad de las soluciones. La deficiencia que se observa en el AGPR es en su tiempo de ejecución.

En general, en las aplicaciones paralelas se observa que juega un papel fundamental el costo de comunicación, lo cual hace que el rendimiento decrezca significativamente. Resultados más significativos podrían ser obtenidos si las pruebas se realizan sobre plataformas paralelas del mayor grado de acoplamiento y/o tamaño más grandes para los problemas estudiados (para el problema de partición de grafos más nodos y/o particiones. y para el problema del viajero de comercio mas ciudades). También podrían minimizarse los tiempos de comunicación si se logra diseñar un esquema paralelo que aumente la granularidad de los procesos paralelos y reduzca el volumen de comunicaciones entre los procesos.

Referencias Bibliográficas

1. Aguilar, J.L.'Allocation de Tâches l'Équilibrage de Charge et l'Optimisation Combinatoire., PhD thesis. Université René Descartes -Paris V, 1995.
2. A. Colorni, M. Dorigo and V. Maniezzo *Distributed Optimization by Ant Colonies* Proccedings of ECAL91, Paris, France 1991 P 134-142.
3. M. Dorigo and L.M. Gambardella ANT-Q: A *Cooperative Learning Approach to Combinatorial Optimization*, Technical Report 95-01. Université Libre Bruxelles, Bruxelles, Belgium, 1995.
4. M. Dorigo, V. Maniezzo, and A, Colorni *The Ant System: Optimization by colony of cooperating agents* IEEE Transactions on Systems, Man, and Cybernetics-Part B, Vol26, No.1,1996, pp. 1-13.

5. Manezzi *Genetic Evolution of the Topology and Weight Distribution of Neural Networks* IEEE Transaction on Neural Networks, Vol.5, No.1, January 1994, pp. 39-53.

6. J. Aguilar, *Resolution du problème de placement de tâches avec de techniques d'optimisation combinatoires*, Proc. 6ème Recontres Francophones du parallelisme, Lyon, France, 1994.

7. L. Davis *Handbook of genetic Algoritmms* Van. No strand Reinhold, New York, 1991.

8. M. Srinivas and M. Patnaik *Genetic Algoritms: A Survey*, IEEE Computer June 1994 P 17-26.

9. D. Macfarlane, I. East *An investigation of several parallel genetic algorithms*, Proc. 12th Occam User group, Exeter, pp.60-67, 1990.

10. [C. Pettey, M. Leuze, J. Grefenstette, *A parallel genetic algorithm*, Proc. 2nd Intl.Conf. of Genetic Algorithms, Cambridge, USA, 1987.

11. H. Muhlenbein, M. Schmisch, J. Born, *The parallel genetic algorithm as function optimizer*, Parallel Computing, Vol. 17, pp. 619-623, 1991.

12. B. Baran, E. Kaszkurewicz, A. Bhaya, *Parallel Asynchronous Team Algorithms convergence and Performance Analysis*,IEEE Transaction on Parallel and Distributed Systems, Vol. 7, No. 7, 1993.

13. J. Aguilar, E. Gelenbe, *Task Assignment and Transaction Clustering Heuristics for Distributed Systems*, Information Sciences, Information and Computer Science, 1997.

Recibido el 12 de Mayo de 1997

En forma revisada el 13 de Enero de 1998