ESTUDIO DE UN ALGORITMO GENETICO PARALELO

Autores:
Gil de Ita Cesar
Muñoz Hernández Arturo
Rendón Marín Manuel

Institución: Facultad de Ciencias de la Electrónica

Dirección: Av. San. Claudio y 18 sur, C.U., Edif.. 129

E-mail: mrendon@ece.buap.mx

RESUMEN

El presente trabajo muestra un primer estudio de los autores sobre la paralelización de un algoritmo genético. En la primera parte se compara, gráficamente, el comportamiento de un algoritmo genético y de un algoritmo de minimización convencional. En la segunda parte, minimizando una función piloto, se realizan mediciones de tiempo de ejecución de nuestro programa paralelo corriendo en uno y mas procesadores.

Dicha paralelización se hace con rutinas MPI actualmente las mas comunes y mas accesibles.

1. INTRODUCCION

La resolución de problemas mediante algoritmos genéticos se ha vuelto mas cotidiano, por lo que su optimización resulta importante y debido a las características que estos poseen es posible su paralelización. El objetivo de este trabajo es el de optimizar el algoritmo genético sin afectar su naturaleza mediante una paralelización estructurada y regular. Debe ser enfatizado que la forma en que se paralelizó es preliminar y se presenta principalmente para sugerir e ilustrar una posible dirección.

2. DESARROLLO

Para empezar diremos que un algoritmo genético tiene sus raíces en la biología e intenta simular el proceso de evolución en una población de individuos[1] pues es una constante búsqueda del mejor individuo. En nuestro caso este proceso comienza con la definición de una población inicial en una matriz de 100 filas por 33 columnas, en donde cada fila representa un cromosoma perteneciente a un individuo. Cada fila es llamada cromosoma y está compuesta por bits generados aleatoriamente, A cada una de las filas la llamaremos individuos; cabe señalar que el tamaño de estos individuos depende del dominio de las variables de nuestra función[1]; como nuestra función $f(x_1,x_2) = 21.5 + x_1*\sin(4\pi x_1) + x_2*\sin(20\pi x_2)$ depende de un x_1 el cual esta definido sobre $-3 \leq x_1 \leq 12.1$ y x_2 que esta definido sobre $4.1 \leq x_2 \leq 5.8$; x1 y x2 tendrán una resolución de 18 y 15 bits respectivamente; sabiendo el tamaño del individuo que es 18+15 =33 bits podemos definir ahora el tamaño de nuestra población que para nuestro caso la definiremos de un tamaño de 100 individuos.

Después de definida nuestra población pasamos a evaluar a cada individuo con nuestra función lo que nos resultará en un vector de 100 números con el resultado de la evaluación de cada individuo, con esto definimos un parámetro muy importante el fitness, que no es otra cosa que la evaluación de nuestra función en un individuo. Después de haber evaluado a toda nuestra población seleccionamos un nuevo grupo de individuos que serán la nueva población con respecto a la distribución de probabilidad de acuerdo al fitness usado; es importante mencionar que el éxito de un algoritmo genético está determinado por el fitness de la función a optimizar, de tal forma que se pueden elegir distintas maneras para formar la nueva población[1], una vez que se ha completado el proceso de evaluación y selección el siguiente paso es aplicar el operador de *crossover* [1] a la nueva población y por último se aplica a los resultados obtenidos del paso anterior el operador de mutación[1]; de tal forma que la estructura del ciclo de algoritmo genético es la siguiente:

1. Evaluación
2. Selección (ruleta)
3. Crossover(cruza)
4. Mutación

El cual se identifica como un algoritmo genético típico [1], que para obtener el individuo óptimo, en este caso el que posee una evaluación máxima, el numero de individuos y el número de generaciones deben tender a un número muy grande [1].

Es notorio que la programación de éste algoritmo en un sistema de cómputo es necesaria debido al calculo que se debe realizar repetidamente para cada ciclo del algoritmo. Así pues una vez programado el algoritmo en lenguaje C bajo ambiente LINUX y al ejecutar el programa con un número de individuos igual a 100, en el primer ciclo del algoritmo que es la creación aleatoria de la población , se obtiene la gráfica de la Figura 1.

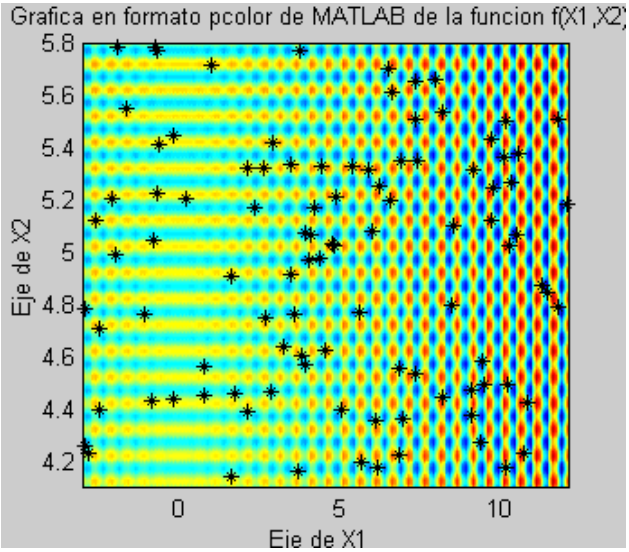


Figura 1

La Figura 1 nos indica la posición de los individuos de la población inicial con estrellas negras, los valores más altos de la función se encuentran en la parte superior derecha de la gráfica en las áreas de color rojo. En el ciclo numero 26 del algoritmo, se obtiene la gráfica de la Figura 2:

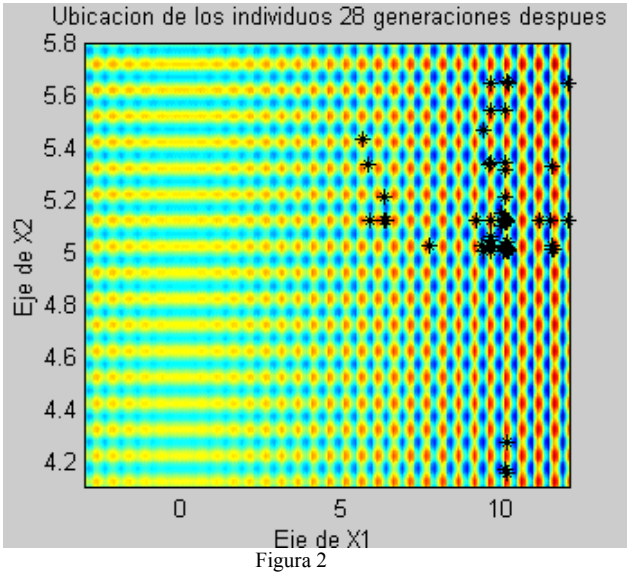
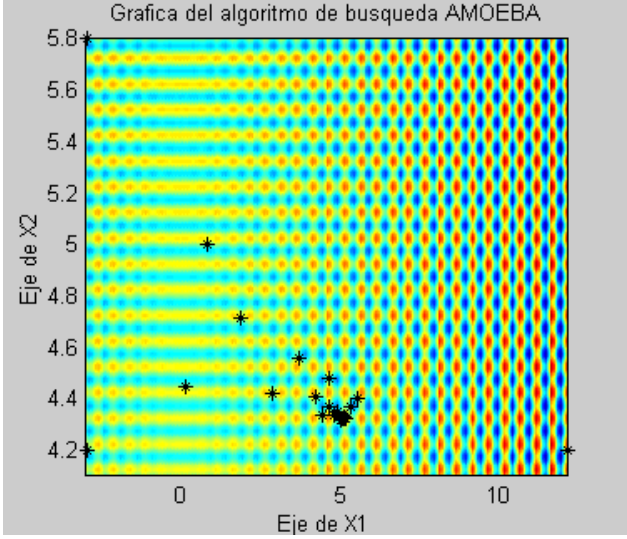


Figura 2



La Figura 2 nos da una idea de la versatilidad de búsqueda de un algoritmo genético, pues como se observa, los resultados tienden a obtener el máximo global conforme las generaciones pasan, en contraposición de un algoritmo de búsqueda serial, el cual obtiene un máximo local y no logra llegar jamás al máximo global, la gráfica siguiente muestra los resultados finales de un algoritmo de este tipo denominado *amoeba*[2], y como se observa, al llegar al final del algoritmo, éste se ha quedado en un máximo local, mientras que en el algoritmo genético conforme se aumenta el numero de generaciones, el resultado final tiende hacia el máximo global.

Figura 3

Al comparar los resultados notamos claramente una superioridad de los algoritmos genéticos con respecto a un algoritmo de búsqueda convencional, esto nos obliga a la optimización de los algoritmos genéticos, por lo que se implementa su paralelización, pues la necesidad de evaluación de cada uno de los individuos en cada una de las generaciones requiere de un tiempo de procesamiento, lo que nos hace pensar de inmediato en la posibilidad de tener varios procesadores que nos permitan la evaluación de forma más rápida de la población.

En el proceso de paralelizar nuestro algoritmo genético saltan a la vista dos posibles formas, una de ellas es dejar a cada procesador operar independientemente en subpoblaciones aisladas y periódicamente compartir sus mejores individuos con otros procesadores a través de un proceso de migración[3] la otra es dejar a cada procesador una parte del algoritmo ya sea la evaluación, crossover, mutación, etc.

En la primer forma cada procesador independientemente genera su propia población de individuos, cada procesador entonces maneja la evaluación del fitness, selecciona los individuos desde una subpoblación para ser usados y generar una nueva en la próxima generación y ejecuta el crossover y mutación sobre esas subpoblaciones. Después de varias generaciones el procesador comparte sus mejores individuos con otros procesadores. El proceso de migración es responsable de varias tareas necesarias para

implementar el intercambio de individuos entre subpoblaciones, cada tarea incluye: seleccionar los emigrantes, enviar los emigrantes, recibir los inmigrantes e integrar a los inmigrantes.

Introducir la migración también nos hace pensar en comunicación en general entre procesos y cuando la comunicación surge el tiempo de esta comienza a dominar el tiempo efectivo de procesamiento, por lo que el desempeño del algoritmo paralelo se ve afectado. Así pues la frecuencia y volumen de información comunicada entre procesos debe ser considerada.

La Estructura de paralelización utilizada en el presente trabajo, respeta los principios de los algoritmos genéticos, pues se aplica el operador de crossover en un solo proceso, es decir, en la estructura de paralelización anterior, es necesario implementar un operador adicional que es la migración, pero esto lo pone en desventaja pues cada individuo debe tener la misma probabilidad de cruza con cualquier individuo de la población, lo que no sucede pues en esta estructura, la población se ha divido en grupos. La presente propuesta evita el problema anterior, pues el operador de selección lo realiza un solo proceso que posee la población completa.

En la paralelización del algoritmo genético se utiliza MPI para realizar la comunicación entre procesadores o procesos, de éste modo, el algoritmo inicia cuando el proceso 0 crea la población inicial, en el siguiente paso que es la evaluación, se paraleliza por lo que un N numero de procesos evalúan un número entero de individuos determinado por la siguiente función $indpproc = \frac{individuos}{procesos}$.

Una vez que el proceso 0 ha repartido a cada procesador restante un grupo de indpproc individuos, éstos, incluyendo el proceso 0 evalúan éste grupo de individuos observe que el proceso de obtención de el fitness de la función se reparte a cada uno de los procesos de tal manera que cada uno ejecuta la evaluación de los individuos.

Cuando se tiene la evaluación de los individuos, ésta es enviada al proceso 0, de tal manera que al final del proceso de comunicación, éste (el proceso 0) tiene todas las evaluaciones de los individuos en un vector de una columna y de tantos renglones como individuos; una vez que el proceso 0 tiene todas las evaluaciones se prosigue a aplicar el operador de la ruleta y el crossover, todo esto es realizado por el mismo proceso 0, y cuando ha terminado y ha obtenido la nueva matriz de individuos después de aplicados los operadores antes mencionados, procede a repartir un grupo de indpproc individuos a cada proceso para que posteriormente cada uno de ellos aplique el operador de mutación a su grupo de individuos y con esto terminar un ciclo del algoritmo.

Al término de cada ciclo cada uno de los procesos incluido el proceso 0 tiene un grupo de indpproc individuos, al que en el paso anterior ha aplicado el operador de mutación, el paso siguiente es iniciar el siguiente ciclo del algoritmo que consiste en nuevamente en evaluar, seleccionar(ruleta), cruzar(crossover) y mutar los individuos de la matriz, por lo que el ciclo se vuelve a iniciar y nuevamente cada uno de los procesos realizará la evaluación de los individuos, retornará estos resultados al proceso 0 , éste selecciona y cruza y reparte nuevamente a los procesos un grupo de individuos para realizar la mutación, éstos pasos se encierran en el ciclo del algoritmo por lo que el numero de ciclos es controlable y el numero de ciclos de algoritmo en nuestro estudio es de 50000 para proveer de una amplia gama de posibilidades.

La siguiente figura muestra un diagrama de flujo de cada ciclo del algoritmo y la respectiva intervención de los diferentes procesos.

Figura 5

En la práctica se observa, que el punto del algoritmo donde se requiere de mayor tiempo de procesamiento que es en la evaluación, donde el número de procesadores que realizan la evaluación de un grupo de individuos determina el tiempo total de evaluación de la población completa, de tal manera que si se tiene una población muy grande y un número de generaciones muy alto, este tiempo en comparación con el que emplearía un solo procesador para realizar todo el ciclo, se ve disminuido notablemente.

3. CONCLUSION

Por sus características, los Algoritmos Genéticos son muy apropiados para incorporar la idea de programación paralela, aún cuando no hay una metodología estándar para ello. Así el paralelismo promete poner a nuestro alcance soluciones a problemas muy complejos

REFERENCIAS

[1] Z. Michalewicz, *Genetic Algorithms+Data Structures = Evolution Programs* (U.S.A.: Springer, 1996).

[2] W.H. Press, S.A. Teukolsky, W.T.Vetterling, B.P. Flannery, *Numerical Recipes in C++* (Trumpington Street, Cambridge, United Kingdom, Cambridge University Press, 1992)

[3] B. Wilkinson, *Parallel Computing*: *Techniques & Applications Using Networked Workstations & Parallel Computers* (Prentice Hall, 1998)