El Algoritmo Genético como alternativa a la Programación Dinámica

Carlos Artemio Coello Coello

Escuela de Ingeniería Civil
Universidad Autónoma de Chiapas
Boulevard Belisario Domínguez km. 1081
Apartado Postal 61
C.P. 29000
Tuxtla Gutiérrez, Chiapas
México

Teléfono : **52+(961) 5-03-22**
Fax : **52+(961) 5-05-27**

E-mail : **coello@eecs.tulane.edu**

Araceli de Lourdes Yáñez López

Instituto Tecnológico de Tuxtla Gutiérrez
Carretera Panamericana km. 1080
Tuxtla Gutiérrez, Chiapas
México

Teléfono : **(52)+(961) 2-49-90**

El Algoritmo Genético como alternativa a la Programación Dinámica

Carlos Artemio Coello Coello
Universidad Autónoma de Chis.
Boulevard Belisario Dguez km. 1081
Tuxtla Gutiérrez, Chiapas (México)
coello@eecs.tulane.edu

Araceli de Lourdes Yáñez López
Instituto Tecnológico de Tuxtla Gutz.
Carretera Panamericana km. 1080
Tuxtla Gutiérrez, Chiapas (México)

RESUMEN

La programación dinámica es una técnica matemática de optimización muy utilizada en una gran variedad de disciplinas que, sin embargo, adolece de varios problemas entre los que destaca el de la dimensionalidad. El algoritmo genético por su parte, es una técnica de optimización más general que puede no sólo resolver la clase de problemas que suelen dejarse a la programación dinámica, sino que además es capaz de mantenerse estable aún en situaciones en que los espacios de búsqueda son muy grandes. Este trabajo muestra la forma de transformar un problema de programación dinámica en uno que pueda ser resuelto por un algoritmo genético.

Introducción

Richard Bellman [1] desarrolló en los años 50s las ideas básicas de la programación dinámica, postulando el principio de optimalidad que dice:

"Una política óptima tiene la propiedad de que cualquiera que sean su estado y decisión iniciales, las decisiones subsecuentes deben constituir una política óptima con respecto al estado resultante de la decisión inicial."

Matemáticamente el principio de optimalidad puede expresarse como:

$$f_n(S_n) = \max_{d_n} \left[R_n(S_n, d_n) + f_{n-1}(S_{n-1}) \right]$$

n=número de etapas subsecuentes en el proceso
S_n=variable de entrada a la *n*-ésima etapa
d_n=variable de decisión en la *n*-ésima etapa
$f_n(S_n)$=retorno máximo de un proceso con *n* etapas y entradas S_n en la *n*-ésima etapa
$r_n=R_n(S_n,d_n)$=función de retorno de la etapa *n* con entrada S_n y variable de decisión d_n
S_{n-1}=salida de la etapa *n* y entrada a la etapa *n*-1
$f_{n-1}(S_{n-1})$=función de retorno máximo desde las etapas 1 a la *n*-1

Esta ecuación puede interpretarse de la siguiente manera: cada componente de una estructura en serie influye en todas las componentes que le siguen, y como sólo la

última componente es independiente, puede ser suboptimizada independientemente para cada variable de estado de entrada que reciba. Una vez logrado esto, se agrupan las 2 últimas etapas y se suboptimizan en forma independiente para cada variable de estado de entrada posible. Este proceso se continúa hasta que todo el problema haya sido optimizado.

El término *programación* hace alusión a una planificación de actividades, de entre las cuales se determinan aquellas que producen la mejor solución o que optimizan el problema. Por su parte, el término *dinámica* se debe al tipo de problemas en que tuvo sus primeras aplicaciones esta técnica, en los que la variable tiempo indicaba el paso de una etapa a otra (por ejemplo, la determinación de la ruta crítica).

La justificación del uso de la programación dinámica para resolver problemas de optimización proviene de la representación incorrecta de muchos fenómenos reales, como los que se resuelven con la programación lineal o con el cálculo diferencial; es decir, la programación dinámica permite resolver problemas de carácter más general que los que permite la programación lineal, presentando además las siguientes ventajas sobre ésta: (a) se obtiene una solución óptima aun en los casos en que la región de soluciones factibles no sea convexa, y (b) no está restringida a variables continuas.

Las ventajas que presenta sobre el cálculo diferencial son aún mayores, ya que las suposiciones que lo definen resultan tremendamente limitantes para una gran parte de los problemas que se presentan en el mundo real. Los requerimientos de que la función que describe el problema sea continua y con derivadas continuas en todos los puntos, nos impedirán resolver problemas como la expansión de un sistema eléctrico o de vías terrestres, en los que la solución queda en términos de variables enteras que describen funciones no continuas.Estos problemas pueden ser resueltos exitosamente por la programación dinámica, pues lo que esta técnica hace es realizar una búsqueda sistemática a través del conjunto de combinaciones posibles de los valores de las variables que definen una función objetivo del problema considerado, hasta encontrar la solución óptima. Sin embargo, la programación dinámica tiene a su vez algunas desventajas:

Cada problema requiere un análisis particular, y de la formulación correcta depende el fracaso o éxito de la técnica.

La limitación más importante es el número de variables estado (entrada) en cada etapa. Si este número se incrementa mucho, las limitantes computacionales se vuelve sumamente serias, porque la explosión combinatoria no se hará esperar. A este problema Bellman lo bautizó como la "maldición de la dimensionalidad" [2].

Algunas de las aplicaciones que se le han encontrado a la programación dinámica son [3]:

Asignación de Recursos : Maximizar los beneficios totales del uso de recursos asignados para un cierto proyecto.
Programación Cronológica de Producción : Cumplir demandas específicas por período durante un horizonte dado de planeación, de manera que se minimicen los costos totales de producción y de inventario.
Problemas de Ajuste : Encontrar un surtido de tamaños estándar de un producto de manera que se minimice el desperdicio o el costo total de producción.
Operaciones de Proceso de Múltiples Etapas : Un producto debe procesarse en una secuencia preestablecida a través de un cierto número de máquinas, cada una de las cuales realiza diferentes cantidades de procesamiento. El objetivo consiste en maximizar la tasa general de procesamiento de la planta.
Control de Procesos Químicos : Un producto debe someterse a una secuencia de reacciones químicas. El objetivo es minimizar el costo del producto final.

Nociones de Algoritmos Genéticos

A fines de los 60s un investigador de la Universidad de Michigan llamado John Holland desarrolló una técnica de búsqueda basada en los mecanismos de supervivencia del más apto planteados en la teoría de la evolución de Darwin. Su objetivo principal era que las computadoras pudieran aprender por sí mismas. Su técnica fue llamada originalmente "planes reproductivos", pero el nombre **algoritmo genético** se hizo popular tras la publicación de su libro [4] en 1975. Sin embargo, la referencia definitiva al tema ha sido el libro publicado por un discípulo de Holland, llamado David Goldberg [5] en 1989. La operación de un algoritmo genético simple puede ilustrarse con el siguiente segmento de pseudo-código [6]:

```
generar población inicial, G(0);
evaluar G(0);
t:=0;
repetir
      t:=t+1;
      generar G(t) usando G(t-1);
      evaluar G(t);
hasta encontrar una solución;
```

Inicialmente, se genera aleatoriamente la población inicial, que estará constituída por un conjunto de cromosomas, o cadenas de caracteres que representan las soluciones

posibles del problema. A cada uno de los cromosomas de esta población se le aplicará la función de aptitud a fin de saber qué tan buena es la solución que está codificando. Sabiendo la aptitud de cada cromosoma, se procede a la selección de los que se cruzarán en la siguiente generación (presumiblemente, se escogerá a los "mejores"). De esta cruza surgirá el siguiente lote de cromosomas con los que se repetirá el proceso anterior. Adicionalmente, existe un operador de mutación que modificará de vez en cuando un alelo (i.e., un bit de la cadena representativa) de un cromosoma. Este operador garantiza la introducción de nuevo material cromosómico, y la interconexión total de nuestro espacio de búsqueda. El Algoritmo Genético se corre durante un número predeterminado de generaciones, o hasta que la población se haya estabilizado (i.e., cuando todos o la mayoría de los individuos tengan la misma aptitud).

Algunas de las ventajas más notables de los Algoritmos Genéticos con respecto a la programación dinámica son las siguientes:

Es independiente del problema que se desea resolver, lo que significa que no se tiene que reformular para cada caso en particular.

Opera de forma simultánea con un gran número de soluciones, en contraposición con la ejecución secuencial de las técnicas tradicionales. A esta propiedad se le conoce como "paralelismo implícito".

Usa operadores probabilísticos en vez de los típicos operadores determinísticos de las otras técnicas.

Se ha aplicado con mucho éxito a problemas que dan lugar a una explosión combinatoria, sin que esto afecte su eficacia.

No suele quedar atrapado en máximos o mínimos locales como las técnicas de búsqueda convencionales.

Ejemplo de Aplicación

A fin de ilustrar cómo resolver un problema típico de la programación dinámica haciendo uso del algoritmo genético, mostraremos a continuación un ejemplo.

Ejemplo 1[1] : Tres plantas térmicas abastecen un centro de carga que demanda una cierta cantidad de energía eléctrica **P** de 180 MW, y se pregunta cuánta energía eléctrica debe proporcionar cada una de las plantas de manera que el costo total de generación sea mínimo, teniendo en cuenta que a cada planta le corresponde una curva característica que relaciona la producción de energía con su costo, y cuyas ecuaciones se muestran a continuación:

[1] Tomado de [7].

Curva A : $C_1 = 0.3P_1^2 - 18.0P_1 + 738.0$
Curva B : $C_2 = 0.166P_2^2 - 6.66P_2 + 616.0$
Curva C : $C_3 = 0.208P_3^2 - 12.08P_3 + 733.3$

donde las **C**s representan los costos en pesos/hora, y las **P**s la potencia generada en MW.

1) Representación : Lo primero que necesitamos determinar para poder aplicar el algoritmo genético, es cuál será el esquema a utilizarse para representar las soluciones posibles del problema. En el planteamiento original del problema se tomó un rango de 20 MW a 80 MW, con intervalos de 5 MW. Nosotros tomaremos un rango de 0 MW a 100 MW con intervalos de 1 MW (un conjunto de valores que haría que la programación dinámica consumiera una cantidad enorme de tiempo). De tal forma, tenemos 101 posibles valores para **P**, o sea que necesitamos 7 bits para representarlos (2^7= 128). Dado que hay 3 posibles **P**s, las cadenas tendrán una longitud de 21 bits (3x7 = 21). De acuerdo a lo anterior, un cromosoma típico en nuestra representación será como el mostrado en la Figura 1.

2) Función de Aptitud : Dado que el objetivo es obtener las potencias que sumen 180, y que tengan un costo mínimo, podemos usar la siguiente función de aptitud penalizada:

$$F(x) = \frac{1.0}{(v * 500 + 1) * (C1 + C2 + C3)}$$

donde $\mathbf{C_1}$, $\mathbf{C_2}$ y $\mathbf{C_3}$ son los costos, que se calculan como indicamos anteriormente, y **v** es el valor absoluto de la diferencia entre la suma obtenida y 180. Nótese que se usa el valor recíproco del costo, porque el algoritmo genético siempre maximiza, y en este caso deseamos efectuar una minimización. Debe tenerse cuidado, por supuesto de que $\mathbf{C_1}+\mathbf{C_2}+\mathbf{C_3}$ no sea cero, pues se produciría un error de sistema. Esta situación puede anticiparse con un chequeo antes de efectuar la división, asignando un valor muy bajo (e.g. cero) a la aptitud en caso de que esta situación se presente.

3) Operadores : Se usará una cruza de 2 puntos, la cual se efectúa de la forma que se indica en la Figura 2. La probabilidad que se dará a la misma será del 80%. En cuanto a la mutación, se le asignará una probabilidad baja, tal y como sugiere Goldberg [5], por lo que será del orden del 1%. El tamaño de población manejado para este ejemplo será de 50 cromosomas, y se correrá el algoritmo genético durante 20 generaciones.

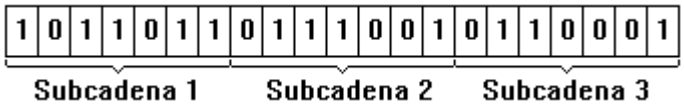

Figura 1 : Cadena representativa de un cromosoma del ejemplo 1.

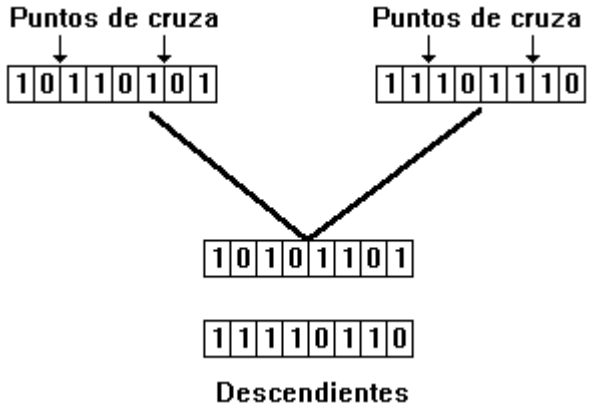

Figura 2 : Uso de 2 puntos de cruza entre 2 individuos. Note cómo en este caso se mantienen los genes de los extremos, y se intercambian los del centro. También aquí existe la posibilidad de que uno o ambos puntos de cruza se encuentren en los extremos de la cadena, en cuyo caso se hará una cruza usando un solo punto, o ninguna cruza, según corresponda.

4) Resultados : El resultado obtenido en una corrida típica es de $2,410.68, correspondiente a los valores de P_1=70 MW, P_2=48 MW y P_3=62 MW. Este valor es menor que el obtenido con programación dinámica usando el planteamiento original del problema, que arroja una solución de $2,623.10. El tiempo que le tomó al algoritmo genético obtener la solución presentada fue de sólo 13 segundos.

Detalles de Implementación

La implementación que usamos para nuestras pruebas es una variante del SGA (Simple Genetic Algorithm) presentado por Goldberg en su libro [5] a la que se le incorporó un mecanismo de selección mediante torneo binario y una cruza de 2 puntos, como se indicó anteriormente. Se mantuvo el esquema de representación binario aconsejado originalmente por Holland, y se manejaron poblaciones pequeñas para todas las pruebas, las cuales se corrieron en una PC AT 286 de 12 MHz, compilándose el código bajo Turbo Pascal 6.0. La salida de resultados se produjo directamente en pantalla, presentándose un reporte resumido por cada generación. Esto implica un ligero retraso en el proceso de cálculo, y la velocidad del algoritmo genético puede incrementarse aún más si nos limitamos a imprimir únicamente la mejor respuesta tras el número de generaciones deseado.

Conclusiones

El problema típico de la programación dinámica es que es víctima de la llamada "maldición de la dimensionalidad" a la que el mismo Bellman hizo alusión. En este trabajo hemos visto cómo, por su parte, la técnica de búsqueda conocida como el algoritmo genético, se ve mucho menos afectada por este fenómeno, además de poder aplicarse de forma más fácil, rápida y eficiente en la solución de un problema de optimización. Si además consideramos que el algoritmo genético tiene un carácter mucho más general, y goza del llamado "paralelismo implícito" que le permite trabajar con varias soluciones simultáneamente, parecen sobrar razones para proponerlo como la mejor opción para resolver problemas de optimización sometidos a restricciones, como los que suelen abordarse con la programación dinámica. Creemos que el uso extendido de esta novedosa técnica se traducirá en grandes beneficios para aquellos profesionales que tienen que enfrentarse a esta clase de problemas de manera cotidiana.

Bibliografía

[1] Bellman, Richard. **Dynamic Programming**. Princeton University Press, Princeton. pp. 83. Nueva Jersey. 1957.

[2] Bellman, Richard. **Adaptive Control Processes: A Guided Tour**. Princeton, New Jersey. Princeton University Press.

[3] Daellenbach, Hans G.; George, John A. y McNickle, Donald C. **Introducción a las Técnicas de Investigación de Operaciones**. CECSA. México. 1987. 711 p.

[4] Holland, John. **Adaptation in Natural and Artificial Systems**. University of Michigan Press. 1975.

[5] Goldberg, David E. **Genetic Algorithms in Search, Optimization and Machine Learning**. Addison-Wesley Publishing Co., Inc. 1989. 412 p.

[6] Buckles, Bill P. & Petry, Frederick E. **Genetic Algorithms**. IEEE Computer Society Press. Technology Series. 1992. 109 p.

[7] Luthe, Rodolfo; Olivera, Antonio y Schutz, Fernando. **Métodos Numéricos**. Editorial Limusa. México. 1990. 443 p.