TOMA DE DECISIONES CON AYUDA DE ALGORITMOS GENÉTICOS.

Eduardo Silva V.

Resumen.

En el presente trabajo se da solución a un problema de optimización clásico por medio de algoritmos genéticos (AG). Se trata de elegir el menor número de elementos que poseen ciertas propiedades, de modo que el conjunto las posea todas.. Para lograr esto, se desarrollaron programas que permiten resolver el tipo de problema tratado en cualquier situación.

Palabras claves: algoritmos genéticos, optimización.

1. Introducción.

En la búsqueda de métodos de optimización eficientes y novedosos, no es difícil dar con la idea de tomar soluciones y hacer combinaciones de ellas con la esperanza de que éstas puedan representar una mejor solución que las primeras. En el marco de lo anterior, los algoritmos genéticos (nombre, en principio, intimidador) surgen como una alternativa válida y sobre todo sugerente dada la simpleza conceptual de sus métodos.

En el presente trabajo, abordaremos la solución de un problema simple de optimización por medio del uso de algoritmos genéticos sencillos (AG). Debemos hacer notar que tanto el problema, como el método son bastante simples, lo que se justifica en los objetivo que perseguimos con este trabajo:

- aprehender los aspectos más básicos y fundamentales de los algoritmos genéticos.
- generar programas computacionales que nos ayuden el la solución de problemas sencillos, usando la metodología mencionada.

2. Marco teórico.

Las ideas y métodos de los AG pueden puntualizarse de la siguiente manera:

1. se genera, en principio al azar, un conjunto de posibles soluciones al problema de interés. Éstas han de codificarse de forma binaria de un modo apropiado. Estas soluciones se denominan **cintas**, y las cintas originales, p**adres.**
2. se genera en base a las soluciones de 1, otro conjunto de posibles soluciones. Esto se logra haciendo combinaciones de las cintas del punto (1). Estas combinaciones o **cruces** se realizan cortando dos las cintas a partir de cierto lugar e intercambiando los trozos resultantes. Se genera, así, un conjunto de **hijos**.
3. luego se evalúa el desempeño de todos los hijos creados y se seleccionan aquellos que den solución al problema. Además, se *marca* a la cinta *óptima*, o sea, a aquella que de solución en mejor forma al problema.
4. a continuación del paso (4), pueden realizarse cambios al azar en los códigos de algunas cintas con el objeto de tratar de dar, por *casualidad*, con mejores soluciones. Este proceso se denomina **mutación**.
5. las cintas seleccionadas en (3), junto con las que resulten *aptas* en el punto 4, conformarán la **nueva** generación de padres. Repitiendo, en base a estas cintas, el proceso desde el punto (1) se pueden obtener mejores resultados que realizando lo anterior sólo una vez.

Debemos notar que lo anterior es sólo un modo de trabajar con AG y que, en principio, no podremos garantizar que con un número bajo de generaciones se llegue a una buena solución si los problemas son complicados.

3. Formulación del problema.

Se consideró un problema de optimización típico, y al cual puede darse solución usando métodos de programación matemática (entera-binaria). En general puede formularse del siguiente modo:

se tiene un número de K elementos (proyectos, entes físicos, etc.), cada uno de los cuales reúne cierto número de características. Hay H características que se estiman deseables y se desea tomar el menor número de elementos de modo que entre todos, se reúnan todas las características.

En consecuencia, se pueden ordenar los elementos y sus características en una matriz M de KxH elementos cada uno de los cuales está caracterizado por:

$$[m_{ij}] = \begin{cases} 1 & si\ el\ elemento\ \ i\ \ tiene\ la\ característica\ \ j \\ 0 & si\ no\ la\ tiene. \end{cases}$$

$$i = 1, 2, ..., K$$
$$j = 1, 2 ..., H$$

Cada cinta está conformada por un vector de K componentes. La i-ésima componente puede tomar dos valores: 0 si la cinta no considera la elección del elemento i, o 1 si la considera.

En base a lo anterior, nuestro problema, en términos de AG, se reduce a encontrar la mejor cinta, o sea, aquella en que la suma sus componentes sea mínima, pero que, además, cumpla con que los elementos a los que hace mención cada una de sus componentes reúnan todas la características.

4. Solución de un caso particular.

En esta sección haremos uso de varios programas que desarrollamos en Matlab (ver apéndice). Los programas son enteramente generales y pueden dar, en principio, solución a problemas de cualquier tamaño.

El problema considerado considera 4 elementos y 5 características. La matriz M queda en este caso :

elemento	características				
	1	2	3	4	5
A	1	0	1	1	0
B	0	0	0	0	1
C	1	1	0	0	1
D	0	1	1	0	0

Posibles soluciones (cintas solución) son, por ejemplo:

sol_1 = [1 1 1 1]
sol_2 = [1 1 1 0]

Sin embargo, no son las mejores pues [1 0 1 0] es claramente la mejor solución. La cinta [1 0 0 0] no es solución ya que el elemento (único de la cinta) A, no posee la característica 2 ni la 5.

Para hacer uso de los programas mencionados basta con ingresar la matriz M y digitar *evolucionar* . El programa pregunta por los dos primeros padres, al menos uno de ellos deberá corresponder a alguna solución. (lo más fácil es considerar $padre_1$ = [1 1 1 1] y $padre_2$ =[0 0 0 0]). Luego se presentan en pantalla los padres de la próxima generación y la cinta óptima (calculada en base a los padres originales). Se da la posibilidad de repetir el proceso en base a la nueva generación.

Debemos mencionar que para garantizar una búsqueda global, se debió incluir mutaciones en todos los miembros de cierta generación, ya que de otro modo el algoritmo tendía a entregar, siempre, una solución que si bien no era mala, no correspondía a la óptima.

4.1 Algunos resultados.

A continuación se muestran los resultados de varias ejecuciones repetidas de los programas, en base a condiciones iniciales (padres) distintas:

a) Primer caso.

padre uno : [1 1 1 1]
padre dos : [0 0 0 0]

nueva generación de padres:

1 1 1 1
1 1 0 1
1 0 1 0

1 1 1 0

óptimo = [1 0 1 0]

b) Segundo caso.

ingrese padre uno : [1 1 1 1]
ingrese padre dos : [0 0 0 0]

nueva generación de padres (segunda):

1 1 1 1
1 1 1 0

óptimo = [1 1 1 0]

se vuelve a repetir en base a los padres generados en la ejecución anterior:

nueva generación de padres (tercera):

1 1 1 1
1 1 1 0
1 1 0 1
1 0 1 0

óptimo = [1 0 1 0]

c) Tercer caso.

padre uno : [1 1 1 1]
padre dos : [0 1 0 0]

nueva generación de padres (segunda):

1 1 1 1
1 1 1 0

óptimo= [1 1 1 0]

se vuelve a repetir en base a los padres generados en la ejecución anterior:

nueva generación de padres (tercera):

1 1 1 1
1 0 1 1
1 1 1 0
1 1 0 1

óptimo = [1 0 1 1]

se vuelve a repetir en base a los padres generados en la ejecución anterior:

1 1 1 1
1 1 1 0
1 1 0 1
1 0 1 0
1 0 1 1

óptimo = [1 0 1 0]

5. Conclusiones:

Los programas utilizados demostraron ser de utilidad, no sólo con el problema presentado en detalle, sino que también con uno que consideró 8 características y 7 elementos, lo que nos permite asegurar que el objetivo planteado fue satisfecho. Por otro lado, se pudo observar lo siguiente:

- independiente de la condición inicial, el AG tiende a la solución óptima (ver caso b y c), lo que permite hacer la solución de los problema independientes de la habilidad del usuario para determinar buenas soluciones como punto de partida.
- otro punto importante es que aunque las condiciones iniciales sean iguales (ver caso a y b), las *rutas* que se sigue para obtener el óptimo son distintas. En el caso (a) bastó con una generación, en el caso (b) se necesitaron dos. Esto es consecuencia de la naturaleza **no** determinística de estos algoritmos de búsqueda. El problema de esto es que en problemas de gran tamaño, es posible que se requieran muchas generaciones para lograr una solución aceptable.
- lamentablemente los tiempos de ejecución de los programas son elevados si se trabaja con muchos elementos, en consecuencia, se deben buscar modos de optimizar los programas usados en este trabajo, si se desea aplicar lo anterior a situaciones reales.

6. Referencias.

Nos basamos, exclusivamente, en una de las clases que don Juan Hernández dictara el primer semestre del año 2001.

Apéndice.

1. Programas usados.

evolucionar

```
clear padre;
clear hjjos;

padre(1).jden=jnput('jngrese
padre uno : ');
padre(2).jden=jnput('jngrese
padre dos : ');

l=0;

whjle (l==0)
   hjjos=nueva_gen(padre);
   [nueva,optjmo]=proxjmos_padres
(hjjos,elem);
   padre(1).jden=nueva(1).jden;
   T=sjze(nueva);
   e=2;
   for j=2:1:T(2)
      z=0;
      for j=e-1:-1:1
      jf
(nueva(j).jden==padre(j).jden)
            z=z+1;
         end
      end

      jf(z==0)

padre(e).jden=nueva(j).jden;
         e=e+1;
      end
   end

   padre.jden
   optjmo

   l=jnput('jngrese 0 para
repetjr (otro número sale): ');
end
```

crear nueva generación apta.

```
functjon hjjos=nueva_gen(padres)

A=sjze(padres);
u_ant=0;

for j=1:1:-1+A(2)
   for j=j+1:1:A(2)
prevjos=reprod(padres(j).jden,pad
res(j).jden);
      U=sjze(prevjos);
      U=U(2);
      for e=u_ant+1:1:U+u_ant
         anterjores(e)=prevjos(e-
u_ant);
      end
      u_ant=U+u_ant;
   end
end

hjjos=anterjores;
```

crear nuevos padres aptos.

```
functjon
[nueva,optjmo]=proxjmos_padres(H,
elem)

A=sjze(H);
B=length(H(1).jden);

k=1;

mejor.jden=ones(1,B);
mejor.apto=1;

for j=1:1:A(2)

H(j).apto=apto(H(j).jden,elem);

   jf (H(j).apto==1)
      candjdato(k)=H(j);
      jf
(sum(candjdato(k).jden)<sum(mejor
.jden))
         mejor=candjdato(k);
      end
      k=k+1;

   end

   jf (H(j).apto==0 |
H(j).apto==1)    %
      U=mutar(H(j));
      U.apto=apto(U.jden,elem);
      jf (U.apto==1)
         candjdato(k)=U;

         jf
(sum(candjdato(k).jden)<sum(mejor
.jden))
            mejor=candjdato(k);
         end
```

```
         k=k+1;

      end
   end

end
```

verifica si una cinta es apta.

```
functjon aptjtud=apto(H,elem)  %
se pasa un hjjo/cjnta

Q=sjze(H);
Q=Q(2);

W=sjze(elem);
W=W(2);

base=zeros(1,W);

for j=1:1:Q
   jf (H(j)==1)
      base = base | elem(j,:);
   end
end

jf (sum(base)==W)
   aptjtud=1;
else
   aptjtud=0;
end
```

mutar

```
functjon mutacjon=mutar(H)

A=sjze(H.jden);

q=random('Djscrete
Unjform',A(2),1,1);

H.jden(q)=not(H.jden(q));
H.apto=0;

mutacjon=H;
```

reproducjr

```
functjon
hjjos=reprod(cjnta1,cjnta2)

L=length(cjnta1);
k=3;
H(k-2).jden=cjnta1;
H(k-1).iden=cinta2;

for i=2:1:L
    previo1=cinta1;
    previo2=cinta2;

     for j=i:1:L

        previo1(j)=cinta2(j);
        previo2(j)=cinta1(j);

        H(k).iden=previo2;
        H(k+1).iden=previo1;
   end
   k=k+2;
end

A=size(H);
for i=1:1:A(2)
   H(i).apto=0;
end

hijos=H;
```