Algoritmo genético para la reducción de ecuaciones de funciones binarias expresadas en maxitérminos y minitérminos.

Carlos Delgado, Edgar Moncada, Luis F. Vargas

Junio de 2012

Resumen

En este proyecto se busca la simplicación de ecuaciones de funciones booleanas utilizando un algoritmo evolutivo, que haga el proceso que se hace con los mapas de Karnaugh

1. Fundamentación del problema

Se busca que en la construcción de circuitos digitales para el cálculo de funciones booleanas sea del menor tamaño posible, existen dos posible formas de representar una función booleana en términos de compuertas *and* y *or*:

- **Minitérminos** Se trata de representar la función en cláusulas, cada clausula es determinada por cada *1* que tiene la función, cada claúsula se crea conectando las entradas con compuertas *and* y las cláusulas entre sí se conectan con compuertas *or*
- **Maxitérminos** Se trata de representar la función en cláusulas, cada clausula es determinada por cada *0* que tiene la función, cada cláusula se crea conectando las entradas con compuertas *or* y las cláusulas entre sí se conectan con compuertas *and*, cada entrada se representa como su negación

Por ejemplo, para la siguiente función:

x_1	x_0	$f(x)$
0	0	1
0	1	1
1	0	0
1	1	0

En representación de minitérminos es así.

$$(\sim x_1 \wedge \sim x_0) \vee (\sim x_1 \wedge x_0) \tag{1}$$

En respresentación de maxitérminos:

$$(\sim x_1 \vee x_0) \wedge (\sim x_1 \wedge \sim x_0) \tag{2}$$

Sin embargo, cuando las funciones tienen muchas variables de entrada, el costo (número de compuertas) presenta un gran incremento, por lo que se deben simplificar las funciones, existen dos métodos:

- **Algebra de Boole** Mediante el uso de relaciones lógicas se simplifican las expresiones, sin embargo, no es bueno cuando las funciones son extensas, ya que se torna engorroso.
- **Mapas de Karnaugh** Una buena solución, ya que utiliza una representación de matrices ordenadas en codificación Grey (cada símbolo cambia un bit en cada columna o fila) y permite asociar directamente para realizar la simplicación, éste metodo es muy bueno hasta cierto tamaño donde las matrices ya son muy grandes y requiere algún esfuerzo para simplificar el problema.

-	x_1	$\sim x_1$
x_0	1	1
$\sim x_0$	0	0

Para la ecuación, en minitérminos:

$$(\sim x_1 \wedge \sim x_0) \vee (\sim x_1 \wedge x_0) \tag{3}$$

En maxitérminos:

$$(\sim x_1 \vee \sim x_0) \wedge (\sim x_1 \vee x_0) \tag{4}$$

2. Algoritmo Genético

2.1. El cromosoma

El cromosoma se codifica como una matriz de tamaño número de clausulas (que es un número entre 1 y número de variables que es el peor caso donde existe una clausula por cada variable) por 2 veces el número de variables de entrada, por ejemplo:

x_1	x_0	$f(x)$
0	0	0
0	1	1
1	0	1
1	1	0

Un cromosoma que representa esta función es:

x_1	$\sim x_1$	x_0	$\sim x_0$
0	1	1	0
1	1	0	0
0	0	1	0

Que en representación de minitérminos:

$$(\sim x_1 \wedge x_0) \vee (\sim x_1 \wedge x_1) \vee x_0 \tag{5}$$

Que en representación de maxitérminos:

$$(\sim x_1 \vee x_0) \wedge (\sim x_1 \vee x_1) \wedge x_0 \tag{6}$$

2.2. Población inicial

Para el problema se utilizan poblaciones iniciales de 200 individuos por defecto, aunque el usuario puede indicar un número, que debe ser mínimo 100 y máximo 1000

2.3. Función de aptitud

La función de aptitud considera los siguiente factores:

- **Número de cláusulas** Número de cláusulas en el cromosoma, se busca minimizar este valor.
- **Acercamiento a la función** Es el factor más importante se busca que este valor sea 0, es decir que la función encontrada sea correcta

Se busca que la función de aptitud sea 0, es decir los mejores cromosomas son aquellos que tengan el menor valor de función de aptitud.

2.4. Función de selección

En este caso se utiliza la selección por ruleta.

2.5. Cruce

Se seleccionan dos cromosomas dentro del grupo de seleccionados, se toma el cromosoma con menor número de cláusulas y se genera un número aleatorio entre 1 y ese número menos 1, valor que se denota con α.

Con éste valor se generan dos hijos, uno tomando en ambos cromosomas α cláusulas iniciales y generando un cromosoma de tamaño $2 * \alpha$, con el resto de ambos cromosomas se realiza el mismo procedimiento

Ejemplo:

Tabla 1: Cromosoma A

x_1	$\sim x_1$	x_0	$\sim x_0$
0	1	1	0
1	1	0	0
0	0	1	0

Tabla 2: Cromosoma B

x_1	$\sim x_1$	x_0	$\sim x_0$
0	0	1	0
1	1	1	0

Como B tiene 2 cláusulas por lo tanto α solamente puede tomar el valor de 1, por lo tanto, los hijos de estos dos padres son:

Tabla 3: Hijo 1

x_1	$\sim x_1$	x_0	$\sim x_0$
0	1	1	0
0	0	1	0

Este hijo se genera al combinar la cĺáusula 1 del hijo A con la claúsula 1 del hijo B

Tabla 4: Hijo 2

x_1	$\sim x_1$	x_0	$\sim x_0$
1	1	0	0
0	0	1	0
1	1	1	0

Este hijo se genera al combinar el resto de cláusulas del hijo A con el resto de cláusulas del hijo B.

2.6. Mutación

Para mutar se selecciona el 2 % de los individuos, en éstos se selecciona aleatoriamente una cláusula, con probabilidad del 50 % se realiza alguna de estas dos acciones

- **Borrar cláusula** Borra una clausula, si es la única de la función, la salida de la misma es siempre 0
- **Cambiar el valor de una variable** Se selecciona una posición de la cláusula y se cambia el valor que tiene asignado por su negación

Si dado el caso se llega seleccionar borrar clausula en un cromosoma con una sóla clausula se aplica la segunda que es cambiar el valor de una variable, para evitar inconsistencias.

2.7. Selección de los cromosomas que pasan a la siguiente generación

En este algoritmo se mantiene el número de la población inicial, por lo tanto para ingresar los hijos se eliminan los peores padres.

2.8. Criterio de parada

El criterio de parada se presenta en dos casos

- Al pasar 200 generaciones
- Si el menor valor de la función de cambio, no tiene cambios en 5 generaciones

3. La aplicación

3.1. Parámetros de entrada

Tabla 5: Parámetros de la aplicación

Parámetro	**Tipo dato**	**Descripción**
-f	Cadena de caracteres	Nombre archivo de entrada. Por defecto *input.txt*
-p	Número natural	Población inicial (Por defecto 200) Min: 100 Máx: 1000
-i	Número natural	Número de interacciones o generaciones (Por defecto 500) Min: 1 Máx: 1000
-t	Booleano (0 o 1)	Indica si se va trabajar por minitérminos o maxitérminos 0 o 1 respectivamente (por defecto 0 o false)
-o	Cadena de caracteres	Nombre archivo de salida. Por defecto *output.txt*

3.2. Representación de la población

La población se representa como un vector de matrices, donde cada matriz representa un individuo.
Ejemplo:

Tabla 6: Representación de población

Individuo	x_1	$\sim x_1$	x_0	$\sim x_0$
Cromosoma 1	0	1	1	0
	1	1	0	0
	0	0	1	0
Cromosoma 2	0	1	1	0
	1	1	0	0
	0	0	1	0
Cromosoma N	0	1	1	0
	1	1	0	0
	0	0	1	0
	0	1	1	0

3.3. La entrada

La entrada se codifica en un archivo de entrada, en cuya primera linea tiene un número natural B que indica el número de bits de la entrada, en las siguiente 2^B líneas se especifica la función booleana, ésta función debe ingresarse en orden de codificación binaria y en la última columna debe tener el valor que toma ante una entrada específica, en la siguiente línea tiene un número natural para indicar el tamaño de la siguiente entrada, sí este número es 0 significa que no hay más entradas.

```
2
0 0 1
0 1 1
1 0 0
1 1 0
3
0 0 0 1
0 0 1 0
0 1 0 1
0 1 1 0
1 0 0 1
1 0 1 0
1 1 0 0
1 1 1 1
0
```

3.4. Parámetros de la aplicación

3.5. Salida

La salida muestra las ecuaciones resultantes de cada solución, la salida tiene la forma solución # <numero de solución>y a continuación la ecuación en términos de x que la representan, ejemplo:

Si es en minitérminos

```
Solución # 1:
(∼x0 and x1) or (∼x1)
Solución # 2:
(∼x0 and x2) or (∼x4) or (x4 and x3)
```

Si es en maxitérminos

```
Solución #1:
(∼x0 or x1) and (∼x1)
Solución #2:
(∼x0 or x2) and (∼x4) and (x4 or x3)
```

4. Análisis y resultados

5. Conclusiones

6. Referencias