Algoritmo Rushdi
Software RMES

Centro de Desarrollo de Gestión Empresarial.
Poniente 1206 – Viña del Mar, Chile.
Fono:(56) (32)688987 – Fax:(56) (32)2684079
empresa@mes.cl

Noviembre, 2009

1. Introducción

Este algoritmo es usado para el cálculo de confiabilidad de sistemas en redundancia o fraccionamiento. Estos sistemas se conocen como k-out-of-n:G, lo cual significa que el sistema funciona sí y sólo sí k de sus n componentes están operativos. La nomenclatura complementaria, k-out-of-n:F, se refiere a que el sistema falla sí y sólo sí k de los n componentes fallan. RMES utiliza la primera definición.

2. El modelo original

La explicación conceptual del algoritmo se encuentra en el documento "The k-out-of-n System Model" en la sección "Symmetric Switching Function Approach by Rushdi", paginas 240-244.

A continuación, sólo explicará lo esencial del algoritmo para que sea de utilidad al momento de implementación y testeo.

La fórmula original esta propuesta para sistemas de tipo k-out-of-n:G. Es recursiva y se expresión es la siguiente:

$$R(i,j) = p_j R(i-1, j-1) + q_j R(i, j-1) \tag{1}$$

El término $R(i,j)$ se refiere a la confiabilidad del sistema i-out-of-j:G, es decir, de un sistema que necesita de que al menos i componentes funcionen para que el sistema no se detenga. Para efectos prácticos, inicialmente sería el sistema en fraccionamiento/redundancia en estudio, en donde i es la cantidad de equipos que deben estar en funcionamiento del total j para que el sistema no falle.

p_j se refiere a la confiabilidad del componente j del sistema.

q_j al complemento de la confiabilidad, $q_j = 1 - p_j$.

La recursión debe tener condiciones de borde para su detención. Estas condiciones son las siguientes:

$$R(0,j) = 1 \tag{2}$$

$$R(j+1,j) = 0 \tag{3}$$

3. Ejemplo

Se calculará la confiabilidad de un sistema 1-out-of-2:G, donde el primer componente tiene confiabilidad 0.8 y el segundo, 0.9. Entonces:

- $p_1 = 0{,}8$
- $p_2 = 0{,}9$
- $q_1 = 0{,}2$
- $q_2 = 0{,}1$

$$\begin{aligned} R(1,2) &= 0{,}9 \cdot R(0,1) + 0{,}1 \cdot R(1,1) \\ R(0,1) &= 1 \end{aligned} \tag{4}$$

$$\begin{aligned} R(1,1) &= 0{,}8 \cdot R(0,0) + 0{,}2 \cdot R(1,0) \\ R(0,0) &= 1 \\ R(1,0) &= 0 \end{aligned} \tag{5}$$

Luego

$$\begin{aligned} R(1,2) &= 0{,}9 + 0{,}1 \cdot 0{,}8 \\ R(1,2) &= 0{,}98 \end{aligned} \tag{6}$$

Finalmente, la confiabilidad del sistema 1-out-of-2:G es de 0.98.

3.1. El algoritmo

El algoritmo más óptimo para la implementación de este cálculo sugiere el uso de una matriz dos dimensional que represente el diagrama de flujo de la figura 1.

La posicion (i, j) representa la confiabilidad $R(i, j)$. Los nodos negros en la primera fila con $i = 0$ son los "nodos fuente" con valores iniciales igual a 1, o sea, $R(0, j) = 1$. Los nodos blancos en la posición $i = j + 1$ son "nodos fuente" con valor inicial 0, o sea, $R(j+1, j) = 0$, $\forall j \geq 0$. Los valores de los otros nodos, por ejemplo, (i, j), se calculan sumando el producto la entrada superior izquierda por p_j al producto de la entrada izquierda por q_j.

El algoritmo actualmente implementado es el siguiente:

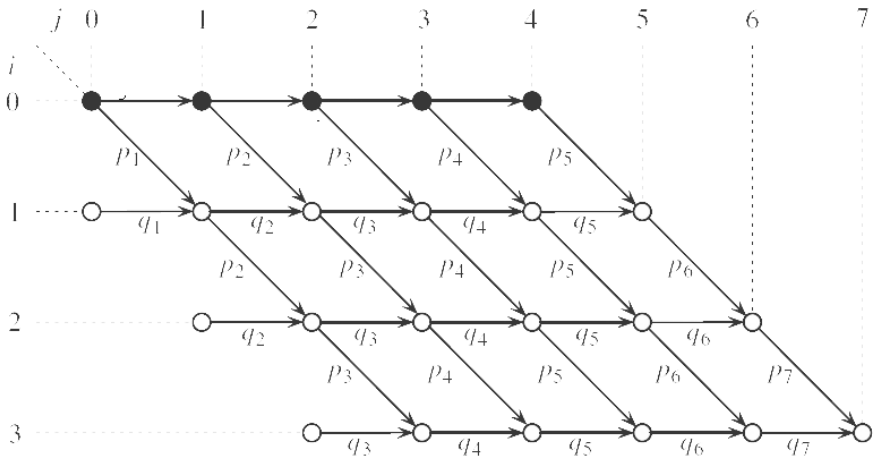

Figura 1: Diagrama de flujo obtenido para R(3,7)

```
double array[][] = new double[k + 1][n + 1];

for (int j = 0; j < n; j++) {
  array[0][j] = 1.0;
}

for (int i = 1; i <= k; i++) {
  for (int j = i; j <= n; j++) {
      double p = ss.getSubsystem(j - 1).R(t);
      array[i][j] = p * array[i - 1][j - 1] + (1 - p) * array[i][j - 1];
  }
}
return array[k][n];
```

Se puede observar que el valor de retorno es el del ultimo nodo de la matriz. Los valores k y n representan al sistema k-out-of-n:G. La variable p almacena la confiabilidad del sistema hijo $j-1$. Esta confiabilidad es la calculada en el tiempo t. También es posible utilizar la confiabilidad por defecto que se calcula con un t igual al tiempo de operación del sistema.