COMPARACIÓN DE ALGORITMOS DE EVALUACIÓN DE CONFIABILIDAD DE SISTEMAS PARCIALMENTE REDUNDANTES

ING. MANUEL A. RIVERA R.

Trabajo de Grado presentando ante la ilustre
Universidad Central de Venezuela para
optar al Título de Magister Scientiarium
en Investigación de Operaciones.

Caracas, Febrero de 2003

© Manuel A. Rivera R. 2003

Hecho el Depósito Legal

Depósito Legal No. lft48720035101

AGRADECIMIENTOS

A mis padres y hermanos por apoyarme incondicionalmente en cada nuevo reto que me propongo en la vida.

A mis panas Miriam, Leopoldo, Yashna, Oscar, Miguel y Carlos por acompañarme y ayudarme en esta fase de mi vida profesional.

A los Prof. Torres y Rocco por darme una oportunidad cuando no la había y por mostrarme nuevos horizontes y posibilidades

A la UCV...

A Julio Verne, Isaac Asimov, Otrovagomas, Neo y Yoda

RESUMEN

En el presente trabajo se han analizado las principales características, funciones de costo y algoritmos de evaluación de confiabilidad más eficientes de SPR, así como también se efectúo una comparación de SPR en cuanto a confiabilidad y costo. En particular se describieron los sistemas K/N:G, CK/N:F, CCK/N:F y LCCS, los cuales constituyen los SPR más estudiados dentro de la literatura.

En los SPR no ponderados y LCCS biestado, los tiempos de ejecución de los algoritmos de evaluación de confiabilidad WCH01, WCH02, KP01, WCH03, WCH04, CCH01, ZT01 y KP02 dependen linealmente de los parámetros N y K del sistema. Por otro lado, los tiempos de ejecución de los algoritmos son semejantes ya que la estructura de los algoritmos es similar y requieren la misma cantidad de recursos de la computadora.

Por todo lo anterior se concluye que un análisis comparativo de tiempos de ejecución para los algoritmos de SPR no ponderados y para los sistemas LCCS biestado no es relevante, recomendando por lo tanto los siguientes algoritmos:

WCH02 $\rightarrow$ Sistemas K/N:G ponderados y no ponderados

WCH03 $\rightarrow$ Sistemas CK/N:F ponderados y no ponderados

CCH01 $\rightarrow$ Sistemas CCK/N:F ponderados y no ponderados

KP02 $\rightarrow$ Sistemas LCSS componentes biestado y multiestado

El análisis comparativo de tiempos de ejecución de los algoritmos para SPR ponderados y LCCS multiestado se deja para cuando se propongan en la literatura nuevos algoritmos para estos sistemas.

A partir del análisis cualitativo efectuado sobre los SPR se puede determinar que los sistemas K/N:G, CK/N:F y CCK/N:F son una interesante alternativa de diseño con relación a los sistemas serie y paralelo, ya que estos sistemas pueden resultar más

confiables que los sistemas serie y menos costosos que los sistemas paralelos. En particular, en los sistemas CK/N:F y CCK/N:F, un simple cambio en la posición de dos componentes puede generar un sistema más confiable y menos costoso.

Con relación a los sistemas LCCS se analizó el efecto que se produce en la confiabilidad y el costo al variar la capacidad de los transmisores pero no se estudió el efecto de la reubicación de componentes.

A partir del modelo general de **[SP]** se proponen los modelos particulares de costo que se emplearon en el análisis cualitativo de los sistemas K/N:G, CK/N:F, CCK/N:F y LCCS.

Se recomienda integrar los algoritmos WCH02, WCH03, CCH01 y KP02 de SPR ponderados y sistemas LCCS multiestado a subrutinas de algoritmos que evalúen la confiabilidad de sistemas complejos ponderados o con restricciones de capacidad.

Con la finalidad de abarcar mayor cantidad de situaciones y casos reales el análisis de confiabilidad y costo efectuado en este trabajo debe ser complementado con análisis de disponibilidad de SPR.

INDÍCE

LISTA DE FIGURAS

LISTA DE TABLAS

GLOSARIO

N Número de componentes del sistema

K 1) Número mínimo de componentes funcionando que hacen que el sistema funcione (sistemas K/N:G no ponderados)

2) Capacidad total mínima requerida para que el sistema funcione (sistemas K/N:G ponderados)

3) Número mínimo de componentes consecutivos fallados que causan la falla del sistema (sistemas CK/N:F y CCK/N:F no ponderados)

4) Capacidad total mínima de componentes consecutivos fallados los cuales causan la falla del sistema (sistemas CK/N:F y CCK/N:F ponderados)

$\mathbf{K_i}$ Número máximo de componentes inmediatamente sucesivos que el componente i puede alcanzar directamente (sistemas LCCS)

$\mathbf{W_i}$ Capacidad del componente i

INTRODUCCIÓN

ANTECEDENTES

En muchas situaciones prácticas, nos debemos enfrentar con sistemas complejos cuya confiabilidad no podemos determinar de manera directa. Si adicionalmente estos sistemas complejos son grandes o moderadamente grandes (como en el caso de sistemas de telecomunicaciones, redes de transmisión y distribución de energía eléctrica, oleoductos, gasoductos, entre otros.) entonces nos encontramos con problemas que sólo son adecuadamente manejados por algoritmos computarizados.

No existe hasta los momentos el algoritmo computarizado que sea efectivo para todas las situaciones que se puedan presentar, de hecho ningún algoritmo computarizado desarrollado para sistemas complejos existente en la literatura consultada tiene la posibilidad de trabajar con todas las variantes de *Sistemas Parcialmente Redundantes* (SPR), los cuáles funcionan si al menos una cierta cantidad de sus componentes funciona.

A partir de los años ochenta, los SPR han demostrado su utilidad en una cada vez mayor gama de situaciones de estudio. Por ejemplo éstos han modelado exitosamente Sistemas de Ignición de Misiones Espaciales **[NO]**, Diseño de Puentes **[PH]**, Sistemas de Telecomunicaciones, Redes de Computadoras **[SP]**, Estaciones Repetidoras de Microondas, Gasoductos/Oleoductos **[CFK]**, entre otros.

Más recientemente –desde principios de los noventa- los SPR han sido objeto de un estudio concienzudo que abarca tópicos como análisis de disponibilidad y mantenibilidad, optimización topológica sujeta a restricciones de costo y confiabilidad, modelación de fallas humanas, etc.

JUSTIFICACIÓN

En la medida que las empresas se encuentran en un mundo signado cada vez más por procesos como la apertura a la libre competencia, la globalización, entre otras, para sobrevivir necesitarán ofrecer a sus clientes, un servicio más barato y confiable. En este contexto, la evaluación de confiabilidad surge como una herramienta primordial dentro

de la planificación, diseño y operación de todos los sistemas que deban modelarse dentro de las empresas.

Se deduce entonces que un estudio de comparación de algoritmos para evaluar la confiabilidad de SPR que complemente los algoritmos desarrollados para sistemas complejos -los cuales no trabajan con SPR como ya se mencionó previamente- puede ser de utilidad para el analista de sistemas que determine que un sistema se deba modelar como un SPR.

ALCANCE Y LIMITACIONES

En el presente trabajo se estudiarán de manera razonablemente amplia aquellos algoritmos de evaluación de confiabilidad relacionados con los conceptos de Conectividad y Capacidad para SPR, incluyéndose un análisis topológico de SPR con consideraciones de costo.

Se dejan para futuros trabajos aquellos otros tópicos relacionados con los SPR que no serán considerados dentro de este estudio, como los conceptos de disponibilidad y mantenibilidad, modelación de errores humanos, simulación, etc.

OBJETIVO GENERAL

Efectuar un análisis cualitativo de los algoritmos para evaluación de confiabilidad de SPR reconocidos dentro de la literatura especializada como eficientes, programar estos algoritmos y suministrar una herramienta con la cual analizar SPR.

ORGANIZACIÓN DEL TRABAJO DE GRADO

El Capítulo I presenta el concepto básico de confiabilidad, las definiciones formales y los modelos de costo de los SPR que analizaremos en este trabajo.

El Capítulo II describe el funcionamiento de los algoritmos que evalúan la confiabilidad de los SPR que mostramos en el Capítulo I, adicionalmente muestra un ejemplo ilustrativo de cada algoritmo.

El Capítulo III presenta el análisis que se efectuó sobre los SPR y los algoritmos de los Capítulos I y II.

Finalmente presentamos las conclusiones y recomendaciones que se derivan de este trabajo.

CAPÍTULO I

CONCEPTOS Y FUNCIONES DE COSTO DE SISTEMAS PARCIALMENTE REDUNDANTES

1.1 CONFIABILIDAD Y EVALUACIÓN DE CONFIABILIDAD

Confiabilidad es la probabilidad que un dispositivo funcione dentro de límites dados al menos durante un período determinado en condiciones ambientales específicas **[MFJ]**. Para efectos de nuestro estudio, los "dispositivos" serán los elementos que componen los sistemas o subsistemas que vamos a estudiar.

Se reconoce en general que existen cuatro tipos genéricos de relaciones estructurales entre un sistema y sus componentes, éstas son: (1) *serie*; (2) *paralelo*; (3) *k/n* y (4) *todas las demás*.

Un *sistema serie* es aquel en el que todos los componentes deben funcionar adecuadamente para que el sistema funcione, por ejemplo el conjunto de hélices de un helicóptero. Un *sistema paralelo* es aquel en que el funcionamiento de cualquiera de los componentes implica el del sistema, por ejemplo el conjunto de dos motores en un avión bimotor. Un *sistema k/n* es uno en el que el funcionamiento de "k" cualesquiera de los "n" componentes del sistema implica el del sistema, por ejemplo el eje trasero de un gran remolque-tractor en el que el funcionamiento de tres de las cuatro ruedas es suficiente para asegurar la movilidad. Un *sistema "todo lo demás"* será uno que posea propiedades y características distintas a los sistemas serie, paralelo y k/n **[NA]**.

La evaluación de confiabilidad de un sistema consiste en determinar la probabilidad de funcionamiento del sistema con base en la probabilidad de funcionamiento de sus componentes. Por ejemplo considere el sistema cuyo Diagrama de Bloques de Confiabilidad (DBC) se representa en la Fig. 1.1:

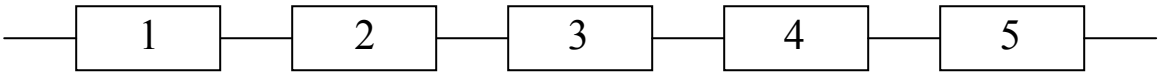


Fig. 1.1 Sistema serie de 5 componentes

Si necesitamos determinar la confiabilidad de este sistema será fácil definirla como el producto de las probabilidades de funcionamiento de los cinco componentes del sistema. En particular nos interesa la evaluación de confiabilidad de sistemas k/n o

parcialmente redundantes como también se les conoce, por ello a continuación presentamos las definiciones formales de estos sistemas.

1.2 SISTEMAS PARCIALMENTE REDUNDANTES

Los Sistemas Parcialmente Redundantes (SPR) no presentan una definición estándar dentro de la literatura especializada, pero si efectuamos una revisión de todo el universo de SPR existente podemos encontrar como característica común la presencia de cierto grado de redundancia en los componentes del sistema, redundancia que no llega a ser total como en los sistemas paralelo pero tampoco se hace nula como en los sistemas serie. En general un SPR funciona si un número predefinido de componentes (no consecutivos, consecutivos, etc.) de los "N" componentes del sistema funcionan.

Los SPR que trataremos en este trabajo son aquellos que encontramos con más frecuencia en la literatura:

a) K/N:G,

b) K/N:F consecutivos lineales o CK/N:F,

c) K/N:F consecutivos circulares o CCK/N:F, y

d) Los sistemas consecutivamente conectados lineales o LCCS.

1.2.1 Sistemas K/N:G

Existe un gran interés en los sistemas K/N:G ya que muchas redes de interconexión (telecomunicaciones, computadoras, sistemas de potencia, entre otras.) pueden ser modeladas con este enfoque **[RSPK]**, así como sistemas tan complejos y críticos como Sistemas de Ignición de Misiones Espaciales **[NO]** y Puentes **[PH]**.

1.2.1.1 Sistemas K/N:G. Concepto de Conectividad.

Un sistema K/N:G se dice que funciona si y sólo si al menos "K" de sus "N" componentes están funcionando. Un sistema K/N:F se dice que falla si y sólo si al menos "K" de sus "N" componentes están fallados **[RSPK]**. La letra "G" es por la palabra

inglesa “Good”, es decir “bueno” o “funcionando” en español, y la letra “F” proviene de “Failed”, es decir “fallado” o “bajo falla” en español. Considere la Fig. 1.2:

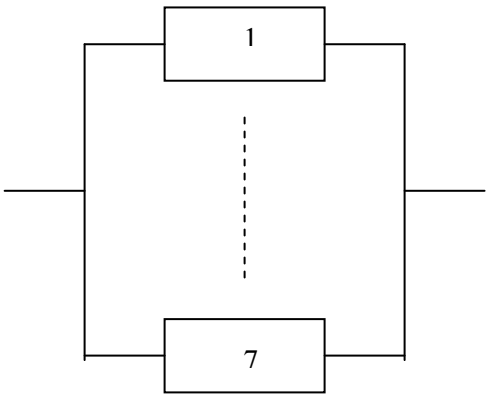


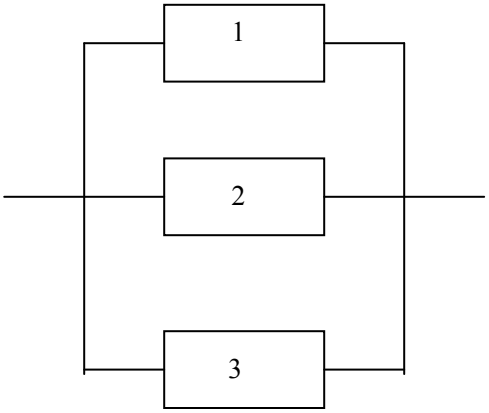
Fig. 1.2 Sistema 4/7:G

La Fig. 1.2 representa el DBC del sistema de calentamiento de un reactor químico formado por 7 resistencias eléctricas, donde por consideraciones de calidad final del producto se dice que el sistema funciona si al menos están operativas 4 de las 7 resistencias disponibles **[WC2]**.

1.2.1.2 Sistemas K/N:G. Concepto de Capacidad

Un sistema K/N:G ponderado es un sistema que consiste de “N” componentes, donde cada uno de los componentes posee un peso o capacidad “W_i”. Este sistema se dice que funciona si y sólo si el peso o capacidad total “W” de los componentes que funcionan es de al menos “K”. El sistema K/N:G es un caso especial del sistema K/N:G ponderado donde el peso o capacidad de cada componente es igual a uno ($W_i = 1$) **[WC1]**. Considere la Fig. 1.3:

Fig. 1.3 Sistema 30/3:G ponderado



La Fig. 1.3 representa el DBC de un sistema de servidores de Internet formado por 3 servidores con las siguientes capacidades de almacenamiento de información, en gigabytes, (GB): servidor 1→ 10 GB; servidor 2 → 20 GB y servidor 3 → 20 GB. Decimos que este sistema funciona si la capacidad de almacenamiento combinada de los servidores en cualquier momento es por lo menos de 30 GB **[WC1]**.

1.2.2 Sistemas CK/N:F

Este tipo de sistemas son relevantes para cierto tipo de redes de telecomunicaciones **[WC3]**, y también pueden modelar sistemas de bombeo y redes de computadora tipo anillo. Dentro de estos sistemas tenemos dos topologías que se estudian con mucha frecuencia: una línea recta (sistemas CK/N:F) y un círculo (sistemas CCK/N:F).

1.2.2.1 Sistemas CK/N:F. Concepto de Conectividad.

Un sistema CK/N:F consiste de una secuencia ordenada de "N" componentes tal que el sistema falla si y sólo si al menos "K" componentes consecutivos fallan **[WC2]**. Considere la Fig. 1.4:

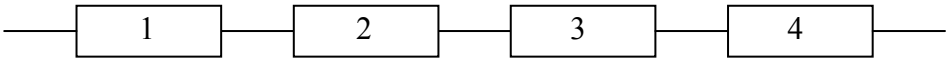


Fig. 1.4 Sistema C2/4:F

La Fig. 1.4 representa el DBC de una máquina de termo-empacado constituida por cuatro hornos eléctricos consecutivos. Se dice que este sistema falla si ocurren fallas de temperatura en por lo menos dos hornos consecutivos de la máquina **[KP]**.

1.2.2.2 Sistemas CK/N:F. Concepto de Capacidad.

Un sistema K/N: F consecutivo lineal ponderado o CK/N:F ponderado consiste de una secuencia ordenada de "N" componentes donde cada uno de los componentes posee un peso o capacidad "W_i". Este sistema se dice que falla si y sólo si el peso o capacidad total "W" de los componentes consecutivos que fallan es de al menos "K" **[WC2]**. El

sistema CK/N:F es un caso especial del sistema CK/N:F ponderado donde el peso o capacidad de cada componente es igual a uno ($W_i = 1$). Considere la Fig. 1.5:

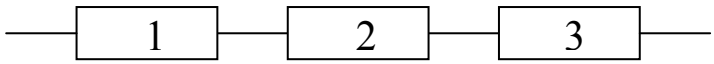


Fig. 1.5 Sistema C3/3:F ponderado

La Fig. 1.5 representa el DBC de un sistema de transporte de agua formado por tres estaciones de bombeo con la siguiente cantidad de bombas por estación: estación 1 → 2 bombas; estación 2 → 3 bombas y estación 3 → 2 bombas. Las bombas son de igual capacidad. El sistema no funciona si al menos no funcionan 3 bombas consecutivas **[WC2]**.

A manera de explicación, supongamos que funcionan las estaciones 2 (3 bombas) y 3 (2 bombas) y falla la estación 1 (2 bombas), en este caso el sistema funciona ya que sólo hay dos bombas consecutivas falladas. Ahora supongamos que funcionan las estaciones 1 (2 bombas) y 3 (2 bombas) y falla la estación 2 (3 bombas), en este caso el sistema falla ya que tenemos 3 bombas consecutivas falladas.

1.2.3 Sistemas CCK/N:F

La definición de este tipo de sistemas es equivalente a la de los sistemas CK/N:F, sólo hay que indicar que los sistemas CCK/N:F se refieren a sistemas de elementos ordenados en forma circular **[CCH]**. Considere la Fig. 1.6:

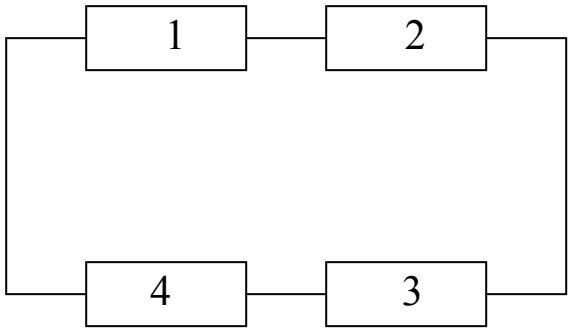


Fig. 1.6 Sistema CC100/4:F ponderado

La Fig. 1.6 representa el DBC de una red de computadoras en las cuales existe una base de datos distribuida. El sistema no funciona si la falla de computadoras consecutivas produce la pérdida de 100 o más datos pertenecientes a la base de datos distribuida **[CCH]**.

1.2.4 Sistemas LCCS

El concepto de sistemas CK/N:F fue extendido por Zuo a los llamados sistemas LCCS **[ZUO]**. Un sistema LCCS consiste de un componente fuente (0), "N" componentes {1,2,...,N} y una carga o sumidero (N+1). La fuente está conectada a los componentes 1, 2,..., K_0, y los componentes j $\{1 \leq j \leq N\}$ están conectados a los componentes $j + 1, j + 2, ..., j + k_j$ por enlaces o arcos. La fuente, la carga y los arcos son perfectos (es decir no fallan), los "N" componentes pueden fallar. El sistema funciona si y sólo si hay una conexión de la fuente a la carga a través de los componentes que estén funcionando. Este tipo de sistemas permite la modelación de sistemas en los cuales los componentes tienen diferentes capacidades de transmisión **[ZUO]**.

Los componentes de los sistemas LCCS sólo pueden estar en uno de dos estados posibles: "funcionamiento" o "falla" y por lo tanto a veces reciben el nombre de componentes biestado. Los autores Kosow y Preuss **[KP]** introdujeron una variante dentro de los sistemas LCCS al suponer que los componentes pueden estar en uno de varios estados posibles (componentes multiestado). Estos sistemas se conocen como LCCMS.

Un sistema LCCMS consiste de N+2 componentes C_i estadísticamente independientes ordenados en una línea ($i \in [0,N]$) y la carga o sumidero C_{N+1} (el cual es perfecto). La falla del sistema es causada por los C_i. Si C_i está en el estado 0 entonces él está fallado, si en cambio C_i está en el estado j $(1 <= j <= K_j$, para un K_j dado) entonces hay caminos hacia los siguientes min(j, N – i + 1) componentes. El sistema falla si y sólo si no hay un camino desde C_o hasta $C_{N+1,}$.

Los sistemas LCCS son un caso particular de los sistemas LCCMS **[KP]**. Considere la fig. 1.7:

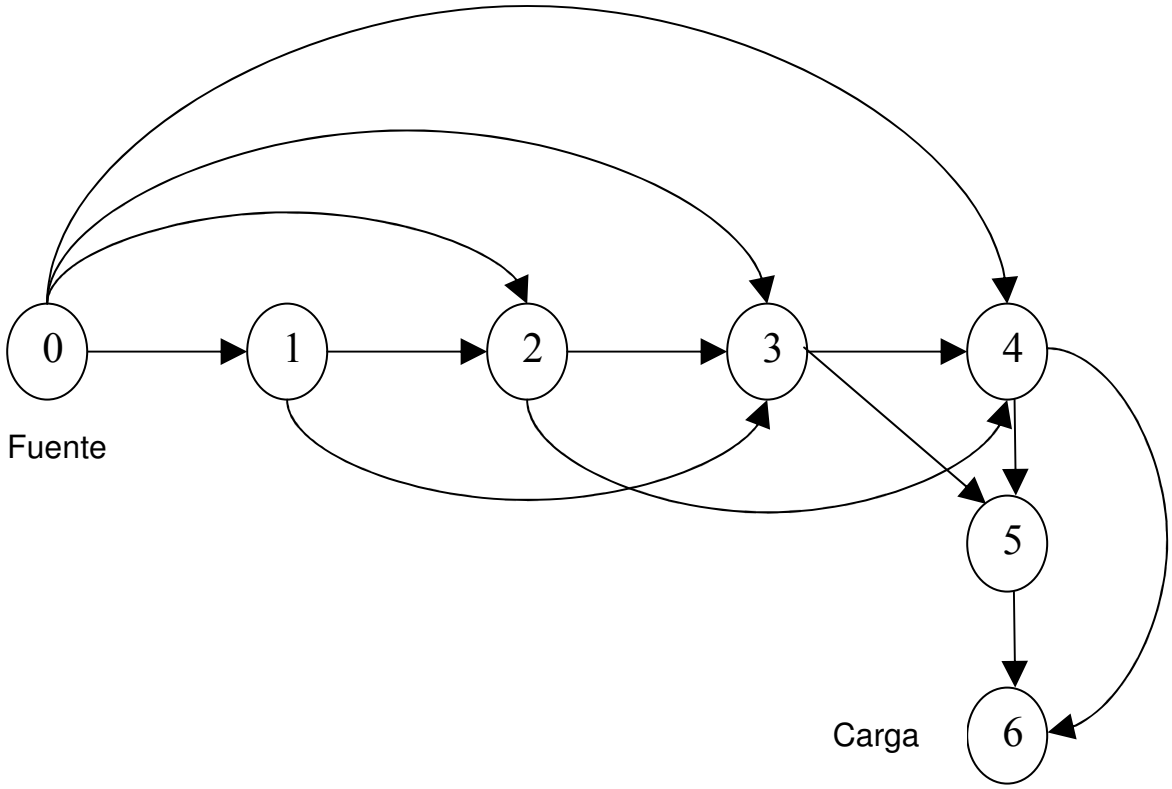


Fig. 1.7 Sistema LCCS de 5 componentes

La Fig. 1.7 representa el DBC de un sistema de telecomunicaciones formado por 5 estaciones repetidoras, una estación transmisora y una estación receptora (perfecta, es decir no presenta falla). La estación transmisora puede transmitir a las estaciones 1, 2, 3 y 4 ($K_0 = 4$), las estaciones repetidoras 1, 2, 3 y 4 pueden transmitir a las dos estaciones más cercanas ($K_1 = K_2 = K_3 = K_4 = 2$), y la estación 5 solo transmite hasta la estación receptora ($K_5 = 1$). El sistema funciona mientras exista un camino entre la estación transmisora y la estación receptora. **[ZUO]**.

Si suponemos que los componentes son *biestado*, cada estación tendrá una probabilidad de funcionamiento y falla asociada, p_i, q_i. Si por el contrario, suponemos que los componentes son *multiestado* cada estación tendrá una probabilidad de falla, q_i, y tantas probabilidades de funcionamiento como posibilidades de transmisión K_i tenga la estación. Por ejemplo, la estación 1 tiene $K_1 = 2$ por lo tanto esta estación tendrá una probabilidad de falla, q_1, y dos probabilidades de funcionamiento, una que

corresponderá a la probabilidad que la estación 1 transmita hasta la estación 2, y otra a la probabilidad que la estación 1 transmita hasta las estaciones 2 y 3.

1.3 FUNCIONES DE COSTO DE SISTEMAS PARCIALMENTE REDUNDANTES

En el diseño de sistemas K/N:G, CK/N:F, CCK/N:F y LCCS necesitamos tanto métodos para evaluar la confiabilidad como herramientas para diseñar de acuerdo con consideraciones de costo. Por ejemplo, cómo escogemos entre un sistema con una confiabilidad de 0,99 y un sistema más costoso pero con una confiabilidad de 0,995? En el proceso de toma de decisiones tenemos que analizar de manera conjunta las funciones de confiabilidad y de costo que representen a los sistemas, análisis que mostramos en el capítulo III **[SP]**. Los modelos de costo los proponemos en este capítulo y los métodos para evaluar la confiabilidad de los sistemas los definimos en el capítulo II.

Para SPR **[SP]** propone la función de costo de la expresión (1.1):

$$C = Cs + E\{\text{ Pérdidas }\} \qquad (1.1)$$

donde:

C	Costo total
Cs	Costo del sistema
E { Pérdidas}	valor esperado de las pérdidas debidas a fallas del sistema

Los supuestos bajos los cuales se basa el modelo son:

1) Los estados de los componentes del sistema son estadísticamente independientes

2) No hay reparación

3) Las operaciones de conexión de los componentes son perfectas, es decir son instantáneas, nunca fallan y no tienen costo.

A partir del modelo de la expresión (1.1) en este trabajo proponemos modelos de costo para los sistemas K/N:G, CK/N:F, CCK/N:F y LCCS, como se detallan a continuación.

1.3.1 Función de Costo de Sistemas K/N:G

Para los sistemas K/N:G proponemos el modelo (1.2):

$$C = NC_1\, g(W) / K + r\, C_2\, [1 - R] \qquad (1.2)$$

donde:

N	número de componentes del sistema
K	factor de redundancia parcial para sistemas K/N:G
W	Capacidad de transmisión de los componentes
C_1	Costo de un SPR con N = 1, K = 1, W = 1 y probabilidad de funcionamiento p.
C_2	Costo por pérdidas causadas por falla del sistema
g(W)	función que depende de la capacidad de transmisión de los componentes
r	confiabilidad del sistema debido a otras causas (errores humanos, fallas eléctricas externas, etc.)
R	confiabilidad del sistema K/N:G

1.3.2 Función de Costo de Sistemas CK/N:F y CCK/N:F

Para los sistemas CK/N:F y CCK/N:F proponemos el modelo de la expresión (1.3):

$$C = NC_1\, g\,(W)\, K + r\, C_2\, [1 - R] \qquad (1.3)$$

donde

K factor de redundancia parcial para sistemas CK/N:F ó CCK/N:F

R confiabilidad de un sistema CK/N:G o CCK/N:F

Los otros términos de (1.3) tienen el mismo significando que en (1.2)

1.3.3 Función de Costo de Sistemas LCCS

Para los sistemas LCCS proponemos el modelo de la expresión (1.4):

$$C = NC_1\, g\,(k_i) + r\, C_2\, [1 - R] \qquad (1.4)$$

donde

K_i capacidades de transmisión de los componentes

$g(k_i)$ función que depende de K_i

R confiabilidad de un sistema LCCS con parámetros N, K y W

Los otros términos de (1.4) tienen el mismo significando que en (1.2) y (1.3)

En el siguiente capítulo presentamos las definiciones formales de los algoritmos de evaluación de confiabilidad de los SPR que mostramos en este capítulo.

CAPÍTULO II

ALGORITMOS DE EVALUACIÓN DE CONFIABILIDAD DE SISTEMAS PARCIALMENTE REDUNDANTES

2.1 INVESTIGACIÓN PRELIMINAR

El autor del presente trabajo efectuó una revisión de los algoritmos de evaluación de confiabilidad de SPR publicados en los últimos veinte años, seleccionando los algoritmos de la tabla 2.1 por constituir los más eficientes y nuevos que reporta la literatura.

Tabla 2.1 Algoritmos de evaluación de confiabilidad de SPR

SPR	ALGORITMO	ALGORITMO
K/N:G	Wu-chen para sistemas K/N:G no ponderados	Wu-Chen para sistemas K/N:G ponderados
CK/N:F	Kosow-Preuss para sistemas CK/N:F no ponderados	Wu-Chen para sistemas CK/N:F ponderados
CCK/N:F	Chang-Chen-Hwang para sistemas CCK/N:F ponderados	Wu-chen para sistemas CCK/N:F no ponderados
LCCS	Kosow-Preuss para sistemas LCCS multiestado	Zuo para sistemas LCCS biestado

En la siguiente sección explicaremos como funcionan estos algoritmos y definiremos los supuestos en los que se basan.

2.2 ALGORITMOS DE EVALUACIÓN DE CONFIABILIDAD DE SISTEMAS PARCIALMENTE REDUNDANTES

2.2.1 Algoritmo Wu-Chen para sistemas K/N:G no ponderados

En 1994 Wu y Chen diseñan un algoritmo que utiliza K procesadores para calcular la confiabilidad de sistemas K/N:G en un tiempo de cómputo de orden N, alcanzando así una velocidad lineal igual a K. Este algoritmo es considerado por **[WC2]** el mejor algoritmo secuencial para evaluar la confiabilidad de sistemas K/N:G.

Supuestos del algoritmo:

1) Cada componente del sistema y el sistema mismo o funciona o falla

2) Los estados de los N componentes son mutuamente independientes.

3) Cada procesador es único.

4) El sistema está funcionando si y sólo si el número de los componentes que funcionan es de al menos K.

Notación empleada:

N	Número de componentes del sistema
p_i, q_i	probabilidad que el componente i funcione (o falle)
K	Número mínimo de todos los componentes funcionando que hacen que el sistema funcione.

Pasos del algoritmo:

Cada procesador contiene los siguientes elementos (ver Fig. 2.1 **[WC2]**):

1) Dos puertos de entrada, I y J

2) Un puerto de salida, O

3) Un buffer, B

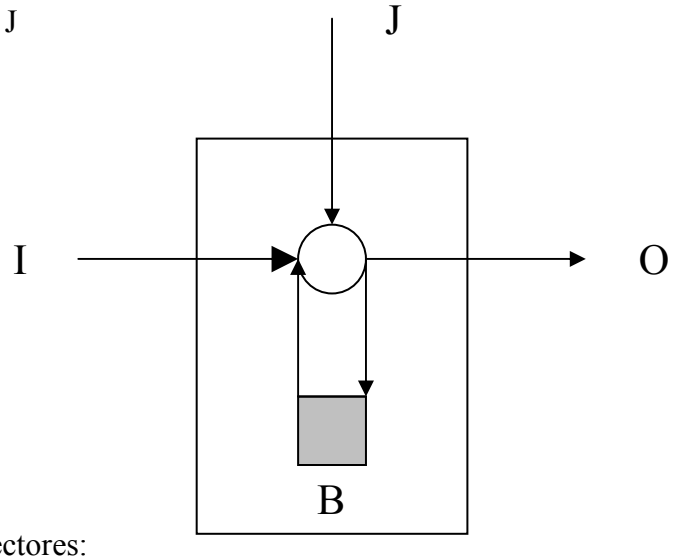


Fig. 2.1 Procesador

Paso 0. Definimos los siguientes vectores:

p[ ] probabilidades de funcionamiento de los componentes

O[ ] salidas

I[ ] , J[ ] entradas

B[ ] buffers

Paso 1. Introducimos y validamos los datos iniciales del sistema.

a) Introducimos N, K, y p_i para i = 1,2,...,N.

b) Verificamos que K y N sean enteros no negativos.

c) Verificamos que $1 \leq K \leq N$.

d) Verificamos que $0 \leq p_i \leq 1$ para todo i.

e) Si no se cumple alguna de las condiciones b), c) o d) volver al paso a)

Paso 2. Inicializamos los buffers de los procesadores.

a) B[0] = 1

b) Desde j = 1 hasta j = K B[j] = 0

Paso 3. Proceso paralelo

Desde i = 1 hasta i = N

{ I[0] = 1

Desde j = 1 hasta j = K

{ O[j] = p[i] I[j-1] + (1-p[i]) B[j]

I[j] = B[j]

B[j] = O[j] }

}

Paso 4. Presentamos los resultados.

El buffer del procesador K, es decir B[K] contiene el valor de la confiabilidad del sistema.

Ejemplo Ilustrativo

Deseamos determinar la confiabilidad del sistema cuyo DBC representamos en la Fig. 2.2. Usaremos el algoritmo Wu-Chen para sistemas K/N:G no ponderados.

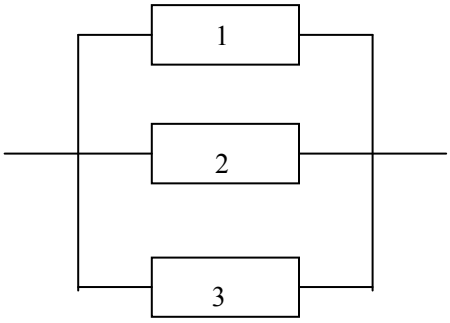


Fig. 2.2 Sistema 2/3:G

Paso 1. Introducimos y validamos los datos iniciales del sistema.

a) Los datos de entrada son N = 3, K = 2 y p_1, p_2 y p_3

b) N y K son enteros no negativos

c) Se cumple que $1 \leq K \leq N$

d) Supongamos que $0 \leq p_i \leq 1$ para todo i.

Paso 2. Inicializamos los buffers.

a) B[0] = 1

b) B[1] = 0 B[2] = 0

Paso 3. Proceso paralelo

Para i = 1

{ I[0] = 1

Para j = 1

{ O[1] = p[1] I[0] + (1-p[1]) B[1]

O[1] = p_1

I[1] = B[1]

B[1] = O[1] }

Para j = 2

{ O[2] = p[1] I[1] + (1-p[1]) B[2]

O[2] = 0

I[2] = B[2]

B[2] = O[2] }

}

Para i = 2

{ I[0] = 1

Para j = 1

{ O[1] = p[2] I[0] + (1-p[2]) B[1]

O[1] = $p_2 + q_2 p_1$

I[1] = B[1]

B[1] = O[1] }

Para j = 2

{ O[2] = p[2] I[1] + (1-p[2]) B[2]

$O[2] = p_2p_1$

I[2] = B[2]

B[2] = O[2] }

}

Para i = 3

{ I[0] = 1

Para j = 1

{ O[1] = p[3] I[0] + (1-p[3]) B[1]

$O[1] = p_3 + q_3(p_2 + q_2p_1)$

I[1] = B[1]

B[1] = O[1] }

Para j = 2

{ O[2] = p[3] I[1] + (1-p[3]) B[2]

$O[2] = p_3(p_2 + q_2p_1) + q_3p_2p_1$

I[2] = B[2]

B[2] = O[2] }

}

Paso 4. Presentamos los resultados.

El buffer del procesador K=2, es decir B[2] contiene el valor de la confiabilidad del sistema cuyo DBC es la Fig. 2.2:

$$p_3(p_2 + q_2p_1) + q_3p_2p_1$$

2.2.2 Algoritmo Wu-Chen para sistemas K/N:G ponderados

Al mismo tiempo que Wu y Chen proponen el algoritmo Wu-Chen para sistemas no ponderados proponen el algoritmo para sistemas ponderados. Este algoritmo es capaz de evaluar la confiabilidad de sistemas que poseen componentes con capacidades diferentes.

Supuestos del algoritmo:

1) Los componentes del sistema y el sistema mismo pueden encontrarse solamente en dos estados posibles, funcionamiento o falla.

2) Los estados de los N componentes son mutuamente independientes.

3) Cada uno de los componentes tiene su peso o capacidad propia.

4) El sistema está funcionando si y sólo si el peso total de los componentes que están funcionando es de al menos K.

Notación empleada:

N Número de componentes del sistema.

K peso o capacidad total mínima de todos los componentes requerida para que el sistema funcione.

W_i Peso o capacidad del componente i.

p_i, q_i Probabilidad que el componente i esté funcionando (bajo falla).

R(i,j) Confiabilidad de un sistema j/i:G ponderado.

Pasos del algoritmo:

Paso 0. Definimos los siguientes vectores y matrices:

p[ ] probabilidades de funcionamiento de los componentes

w[ ] pesos o capacidades de transmisión de los componentes

R[ ][ ] confiabilidad

Paso 1. Introducimos y validamos los datos iniciales del sistema.

a) Introducimos N, K, W_i y p_i para i = 1, ..., N.

b) Verificamos que K, N y W_i, para i = 1, ..., N, sean enteros no negativos.

c) Verificamos que $0 \leq p_i \leq 1$ para todo i.

d) Si no se cumple alguna de las condiciones b) o c) volver al paso a)

Paso 2. Construimos la matriz R

a) Desde i = 0 hasta i = N

R[i][0] = 1;

b) Desde j = 1 hasta j = K

R[0][j] = 0;

c) Desde i = 1 hasta i = N y desde j = 1 hasta j = K

{ Si (j – w[i]) ≥ 0

R[i][j] = p[i] R[i-1][j-w[i]] + (1-p[i]) R[i-1][j]

En caso contrario

R[i][j] = p[i] + (1-p[i]) R[i-1][j] }

Paso 3. Presentamos los resultados.

La confiabilidad del sistema es igual al elemento R[N][K] de la matriz R.

Ejemplo Ilustrativo

Deseamos determinar la confiabilidad del sistema cuyo DBC representamos en la Fig. 2.2. Usaremos el algoritmo Wu-Chen para sistemas K/N:G ponderados.

Paso 1. Introducimos y validamos los datos iniciales del sistema.

a) Los datos de entrada son N = 3, K = 2 , $W_1=W_2=W_3=1$, p_1, p_2 y p_3

b) N, K y W_i, para i = 1, ..., N son enteros no negativos.

c) Supongamos que $0 \leq p_i \leq 1$ para todo i.

Paso 2. Construimos la matriz R

a) Desde i = 0 hasta i = N = 3

R[0][0] = 1; R[1][0] = 1; R[2][0] = 1;

R[3][0] = 1;

b) Desde j = 1 hasta j = K = 2

R[0][1] = 0; R[0][2] = 0;

c) Desde i = 1 hasta i = N = 3 y desde j = 1 hasta j = K = 2

i = 1, j = 1

(j – w[i]) = (1 – 1) = 0 $\Rightarrow$

R[1][1] = p[1] R[0][0] + (1-p[1]) R[0][1]

R[1][1] = p_1

$i = 1, j = 2$

$(j – w[i]) = (2 – 1) > 0 \Rightarrow$

$R[1][2] = p[1]\ R[0][1] + (1\text{-}p[1])\ R[0][2]$

$R[1][2] = 0$

$i = 2 , j = 1$

$(j – w[i]) = (1 – 1) = 0 \Rightarrow$

$R[2][1] = p[2]\ R[1][0] + (1\text{-}p[2])\ R[1][1]$

$R[2][1] = p_2 + (1\text{-}p_2)p_1 = p_2 + q_2p_1$

$i = 2 , j = 2$

$(j – w[i]) = (2 – 1) > 0 \Rightarrow$

$R[2][2] = p[2]\ R[1][1] + (1\text{-}p[2])\ R[1][2]$

$R[2][2] = p_2p_1$

$i = 3 , j = 1$

$(j – w[i]) = (1 – 1) = 0 \Rightarrow$

$R[3][1] = p[3]\ R[2][0] + (1\text{-}p[3])\ R[2][1]$

$R[3][1] = p_3 + (1\text{-}p_3)\ (p_2 + (1\text{-}p_2)p_1)$

$= p_3 + q_3(p_2 + q_2p_1)$

$i = 3$ y $j = 2$

$(j – w[i]) = (2 – 1) > 0 \Rightarrow$

$R[3][2] = p[3]R[2][1] + (1\text{-}p[3])R[2][2]$

$$R[3][2] = p_3(p_2+q_2p_1) + (1-p_3)(p_2p_1)$$

$$= p_3(p_2 + q_2p_1) + q_3p_2p_1$$

Paso 3. Presentamos los resultados.

La confiabilidad del sistema es igual al elemento R[3][2] de la matriz R, por lo tanto la confiabilidad del sistema cuyo DBC es la Fig. 2.2 es:

$$p_3(p_2 + q_2p_1) + q_3p_2p_1$$

Esta expresión es idéntica a la que se obtiene con el algoritmo Wu-Chen para sistemas K/N:G no ponderados, debido a que las capacidades de los elementos son iguales.

2.2.3 Algoritmo Kosow-Preuss para sistemas CK/N:F no ponderados

Kosow y Preuss presentan en 1995 un algoritmo para determinar la confiabilidad de sistemas LCCS el cuál bajo ciertas condiciones (componentes biestado y $K_i = K$ para todo los componentes) se puede utilizar en la evaluación de confiabilidad de sistemas CK/N: F.

Supuestos del algoritmo:

1) Cada componente y el sistema funcionan o están bajo falla
2) Los estados de los N componentes son mutuamente independientes.
3) El sistema falla si y sólo si al menos K componentes consecutivos fallan.

Notación empleada:

N número de componentes del sistema

K número mínimo de componentes consecutivos fallados los cuales causan la falla del sistema

p_i, q_i probabilidad que el componente i funcione (falle)

Pasos del algoritmo:

Paso 0. Definimos los siguientes vectores:

p[ ] probabilidades de funcionamiento de los componentes

q[ ] probabilidades de falla de los componentes

R[ ] confiabilidad

Paso 1. Introducimos y validamos los datos iniciales del sistema.

a) Introducimos N, K y p_i para i = 1, ..., N

b) Verificamos que N y K sean enteros no negativos.

c) Verificamos que $0 \leq p_i \leq 1$ para todo i.

d) Si no se cumple alguna de las condiciones b) o c) volver al paso a)

e) Calculamos q_i a partir de la expresión $q_i = 1 - p_i$, para i =1,...,N

Paso 2. Evaluamos los componentes del vector de confiabilidad R:

Para j = 0 hasta j = K - 1

{ R[j] = 1 }

T = 1

Para r = 1 hasta r = K

{ T = T *q[r] }

R[K] = 1 – T

Para j = K + 1 hasta N

{ R[j] = R[j-1] - p[j-K] * fq * R[j-K-1] }

Cálculo de fq:

fq = 1

Para r = 0 hasta r = K-1

{ fq = fq * q[N-r] }

Paso 3. Presentamos los resultados

La confiabilidad del sistema CK/N:F será igual al elemento R[N] del vector R.

Ejemplo ilustrativo:

Deseamos determinar la confiabilidad del sistema cuyo DBC representamos en la Fig. 2.3. Usaremos el algoritmo Kosow-Preuss para sistemas CK/N:F no ponderados.

Fig. 2.3 Sistema C3/4:F

Paso 1. Introducimos y validamos los datos iniciales del sistema.

a) Los datos de entrada son N=4, K=3 , p_i , p_2 , p_3 y p_4

b) N y K son enteros no negativos.

c) Supongamos que se cumple que $0 \leq p_i \leq 1$ para todo i.

d) Calculamos q_i a partir de la expresión $q_i = 1 - p_i$, para i =1,...,N

Paso 2. Evaluamos los componentes del vector de confiabilidad R:

j = 0 R[0] = 1

j = 1 R[1] = 1

j = 2 R[2] = 1

T = 1

r = 1 $T = T * q[1] = (1)(q_1) = q_1$

r = 2 $T = T * q[2] = (q_1)(q_2) = q_1q_2$

r = 3 $T = T * q[3] = (q_1q_2)(q_3) = q_1q_2q_3$

$R[3] = 1 - T$

$R[3] = 1 - q_1q_2q_3$

j = 4 R[4] = R[4-1] - p[4-3] * fq * R[4-3-1]

R[4] = R[3] - p[1] * fq * R[0]

Cálculo de fq:

fq = 1

T = 0

r = 0 $fq = (1)\ q[4-0] = q_4$

r = 1 $fq = (q_4)\ q[4-1] = q_4q_3$

r = 2 $fq = (q_4q_3)q[4-2] = q_4q_3q_2$

Una vez calculado fq, retornamos al algoritmo principal:

R[4] = R[3] - p[1] * $q_4q_3q_2$ * R[0]

$R[4] = 1 - q_1q_2q_3 - p_1q_2q_3q_4(1) = 1 - q_1q_2q_3 - p_1q_2q_3q_4$

Paso 3. Escribimos los resultados

La confiabilidad del sistema cuyo DBC es la Fig. 2.3 vendrá dada por R(4):

$$1 - q_1q_2q_3 - p_1q_2q_3q_4$$

2.2.4 Algoritmo Wu-Chen para sistemas CK/N:F ponderados

En 1994 Wu y Chen definen los sistemas CK/N:F ponderados y proponen un algoritmo para evaluar la confiabilidad de estos sistemas **[WC3]**.

Supuestos del algoritmo:

a) Cada componente y el sistema funciona o está fallado.

b) Los estados de los N componentes son mutuamente independientes.

c) Cada componente tiene su propio peso positivo entero y el sistema falla si y sólo si el peso total de los componentes consecutivos fallados es al menos de K

Notación empleada:

N	número de componentes del un sistema
p_i,,q_i	probabilidad que el componente i (funcione o falle)
K	peso total mínimo de componentes consecutivos fallados los cuales causan la falla del sistema
W_i	peso del componente i
S_i	evento mínimo que causa que el sistema falle
m	número total de todos los posibles S_i
Beg(i)	primer componente de S_i
End(i)	último componente de S_i

Wet(i)	peso total de S_i
Q(i)	$\prod_{j=Beg(i)}^{End(i)} q_j$ producto de probabilidades de falla de los eventos S_i
$R_L(i,j)$	confiabilidad del sistema lineal consistente de los componentes i, i+1, ..., j
F_L	$1 - R_L$ complemento de la confiabilidad del sistema

Pasos del algoritmo:

Paso 0. Definimos los vectores:

p[ ]	probabilidad de funcionamiento de los componentes
q[ ]	probabilidad de falla de los componentes
W[ ]	pesos de los componentes
Beg[ ] y End[ ]	componentes primero y último de S
Wet[ ]	peso total de S
Q[ ]	producto de probabilidades de falla
F_L[ ]	complemento de la confiabilidad

Paso 1. Introducimos y validamos los datos iniciales del sistema.

a) Introducimos N, K, W_i y q_i para i = 1, ..., N

b) Verificamos que N, K y W_i sean enteros no negativos para todo i.

c) Verificamos que $0 \leq q_i \leq 1$ para todo i.

d) Si no se cumple alguna de las condiciones b) o c) volver al paso a)

e) Calculamos p_i a partir de la expresión $p_i = 1 - q_i$, para i =1,...,N

Paso 2. Encontramos los eventos S_i

```
m = 0                   event = 1       Wet[event] = 0

Q[event] = 1            Beg[event] = 1

Para component = 1 hasta component = n

{       Wet[event] = Wet[event] + w[component]

        Q[event] = Q[event]q[component]

        Si Wet[event] ≥ K

        {       m = m + 1

                End[event] = component

                Mientras (Wet[event] - w [Beg[event] ] ≥ K)

                {       Wet[event] = Wet[event] - w[ Beg[event]]

                        Q[event] = Q[event] / q [Beg[event]]

                        Beg[event] = Beg[event] + 1

                }

                Beg[event + 1] = Beg[event] + 1

                Wet[event + 1] = Wet[event] - w [Beg[event]]

                Q[event + 1] = Q[event] / q [Beg[event]]

                event = event + 1
```

}

}

Paso 3. Evaluamos los FL del sistema

Para j =1 hasta j = End[1] - 1

{ F_L [j] = 0 }

F_L[End[1]] = Q[1]

Para j = 2 hasta j = m

{ H = 0

Para i = 0 hasta i = (Beg[j] - Beg[j-1] - 1)

{ H = H + (1-F_L[Beg[j-1]+i-1]p[Beg[j-1] + i] * H_1 }

Calculo de H_1:

H_1= 1

Para l = (Beg[j-1] + i + 1) hasta Beg[j] - 1

{ H_1 = H_1 * q[l] }

Una vez calculado H_1 retornamos de la subrutina:

F_L[End[j]] = H + F_L[End[j-1]]

}

Para j = (End[m] + 1) hasta N

{ F_L[j] = F_L[End[m]] }

Paso 4. Presentamos los resultados

La confiabilidad del sistema vendrá dada por:

$$R_L[N] = 1 - F_L[N]$$

Ejemplo ilustrativo:

Deseamos determinar la confiabilidad del sistema cuyo DBC representamos en la Fig. 2.3. Usaremos el algoritmo Wu-Chen para sistemas CK/N:F ponderados.

Paso 1. Introducimos y validamos los datos iniciales del sistema.

a) Los datos de entrada son N=4, K=2, $W_i = 1$ para todo i y q_i , q_2, q_3 y q_4

b) N, K y W_i son enteros no negativos para todo i.

c) Supongamos que se cumple que $0 \leq q_i \leq 1$ para todo i.

d) Calculamos p_i a partir de la expresión $p_i = 1 - q_i$, para i =1,...,N

Paso 2. Encontramos los eventos S_i

m = 0 event = 1 Wet[event] = 0

Q[event] = 1 Beg[event] = 1

component = 1

Wet[1] = Wet[1] + w[1] = 0 + 1 = 1

$Q[1] = Q[1]q[1] = (1)q_1 = q_1$

$Wet[1] < K \Rightarrow$

component = 2

Wet[1] = Wet[1] + w[2] = 1 + 1 = 2

$Q[1] = Q[1]q[2] = q_1q_2$

$Wet[1] < K \Rightarrow$

component = 3

$Wet[1] = Wet[1] + w[3] = 2 + 1 = 3$

$Q[1] = Q[1]q[3] = q_1q_2q_3$

$Wet[1] = K \Rightarrow$

$m = m + 1 = 0 + 1 = 1$

$End[1] = component = 3$

$Wet[1] - w\ [\ Beg[1]\ ] < K \Rightarrow$

$Beg[1 + 1] = Beg[1] + 1 = 1 + 1 = 2$

$Wet[1 + 1] = Wet[1] - w\ [Beg[1]] = 3 - w[1] = 3 – 1 = 2$

$Q[1 + 1] = Q[1] / q\ [Beg[1]] = q_1q_2q_3 / q[1] = q_1q_2q_3/q_1 = q_2q_3$

$event = event + 1 = 1 + 1 = 2$

component = 4

$Wet[2] = Wet[2] + w[4] = 2 + 1 = 3$

$Q[2] = Q[2]q[4] = q_2q_3q_4$

$Wet[2] = K \Rightarrow$

$m = m + 1 = 1 + 1 = 2$

$End[2] = component = 4$

Wet[2] - w [Beg[2]] < K $\Rightarrow$

Beg[2 + 1] = Beg[2] + 1 = 2 + 1 = 3

Wet[2 + 1] = Wet[2] - w [Beg[2]] = 3 – 1 = 2

Q[2 + 1] = Q[2] / q [Beg[2]] = $q_2q_3q_4$ / q_2 = q_3q_4

event = 2 + 1 = 3

Paso 3. Evaluamos los F_L

j =0 F_L [0] = 0

j =1 F_L [1] = 0

j =2 F_L [2] = 0

F_L[End[1]] = Q[1]

F_L[3] = $q_1q_2q_3$

Para j = 2 hasta j = m = 2

j = 2

H = 0

i = 0

H = H + (1-F_L[Beg[2-1]+0-1])p[Beg[2-1] + 0] *H_1 * Q[2]

H = H + (1-F_L[0])p[1] *H_1* Q[2]

Calculo de H_1:

H_1= 1 y retornamos al algoritmo principal

H = 0 + (1- 0)p_1(1)$q_2q_3q_4$

$$H = p_1q_2q_3q_4$$

$$F_L[\ End[2]] = H + F_L[\ End[2-1]]$$

$$F_L[\ End[2]] = p_1q_2q_3q_4 + F_L[End[1]] = p_1q_2q_3q_4 + F_L[3]$$

$$= p_1q_2q_3q_4 + q_1q_2q_3$$

$$F_L[4] = p_1q_2q_3q_4 + q_1q_2q_3$$

Paso 4. Presentamos los resultados

La confiabilidad del sistema cuyo DBC es la Fig. 2.3 vendrá dada por $1 - F_L(4)$:

$$1 - F_L(4) = 1 - (p_1q_2q_3q_4 + q_1q_2q_3)$$

$$1 - p_1q_2q_3q_4 - q_1q_2q_3$$

Esta expresión es idéntica a la que se obtiene con el algoritmo Kosow-Preuss para sistemas CK/N:F no ponderados, debido a que las capacidades de los elementos son iguales.

2.2.5 Algoritmo Wu-Chen para sistemas CCK/N:F no ponderados

En 1992 Wu y Chen proponen un algoritmo para evaluar la confiabilidad de sistemas CCK/N:F que según estos autores presenta un tiempo de cómputo de orden NK, mucho mejor que otros algoritmos que presentan tiempos de cómputo de orden NK^2 **[WC3]**.

Supuestos

a) Cada componente, subsistema y sistema funciona o está fallado

b) Los estados de los N componentes son mutuamente independientes.

c) Los componentes 1, 2, ..., N están arreglados en un círculo y en ese orden.

d) El sistema falla si y solo si al menos K componentes consecutivos fallan

Notación empleada

N	número de componentes del sistema
K	número mínimo de componentes consecutivos fallados los cuales causan la falla del sistema
i	índice del componente; i = 1, 2, ..., n
p_i, q_i	probabilidad que el componente i funcione (falle)
sys	sistema CCK/N:F
sys-0	sistema CK/N:F
sys-i	sistemas K/(N + i):F consecutivos lineales, para i = 1, 2, ..., K–1; siendo $q_j = 1$ para j = 1, 2, ..., i, N + 1, ..., N + i
F_{sys}	Pr{sys esté fallado}
F_{sys-i}	Pr{sys-i esté fallado}, para i = 0, 1, …. K –1
F'_{sys-i}	Pr{sys-i esté fallado}, para i = 1, ..., K – 1, donde solo se consideran los primeros (N + i – 1) componentes

Pasos del algoritmo:

Paso 0. Definimos los vectores:

p[ ]	probabilidad de funcionamiento de los componentes
q[ ]	probabilidad de falla de los componentes
W[ ]	pesos de los componentes
Beg[ ] y End[ ]	componentes primero y último de S
Wet[ ]	peso total de S

Q[ ] producto de probabilidades de falla

F_L[ ] complemento de la confiabilidad

Paso 1. Introducimos y validamos los datos iniciales del sistema.

a) Introducimos N, K, y q_i para i = 1, ..., N

b) Verificamos que N y K sean enteros no negativos

c) Verificamos que $0 \leq q_i \leq 1$ para todo i.

d) Si no se cumple alguna de las condiciones b) o c) volver al paso a)

e) Calculamos p_i a partir de la expresión $p_i = 1 - q_i$, para i =1,...,N

Paso 2. Calculamos F_{sys-0}

$H = F_{sys-0}$

Paso 3. Desde i = 1 hasta i = K – 1

{ $T = F_{sys-i} - F'_{sys-i}$

Desde j = 1 hasta j = i

{ T = T q[j] }

H = H + T

}

R = 1 – H

NOTA: Las F_{sys-i} y F'_{sys-i} las determinamos a partir del algoritmo Wu–Chen para sistemas CK/N:F ponderados.

Paso 4. Presentamos los resultados

La confiabilidad del sistema CCK/N:F vendrá dado por el valor de R

Ejemplo ilustrativo:

Deseamos determinar la confiabilidad del sistema cuyo DBC representamos en la Fig. 2.4. Usaremos el algoritmo Wu-Chen para sistemas CCK/N:F no ponderados.

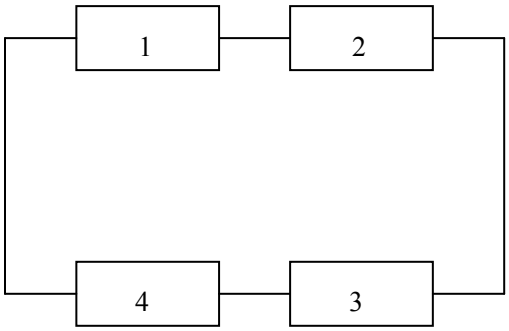


Fig. 2.4 Sistema CC3/4:F

Paso 1. Introducimos y validamos los datos iniciales del sistema.

a) Los datos de entrada son N = 4, K = 3, q_1, q_2, q_3 y q_4

b) N y K son enteros no negativos

c) Supongamos que $0 \leq q_i \leq 1$ para todo i.

d) Calculamos p_i a partir de la expresión $p_i = 1 - q_i$, para i =1,...,N

Paso 2. Calculamos F_{sys-0}

$$H = F_{sys-0} = (p_1q_2q_3q_4 + q_1q_2q_3)$$

NOTA : F_{sys-0} lo determinamos a partir del algoritmo Wu-Chen para sistemas CK/N:F

Paso 3. i = 1

$$T = F_{sys-1} - F'_{sys-1}$$

$$F_{sys-1} = p_2q_3q_4 + q_2q_3 \quad y \quad F'_{sys-1} = q_2q_3$$

NOTA : F_{sys-1} y F'_{sys-1} los determinamos a partir del algoritmo Wu-Chen para sistemas CK/N:F

$$T = p_2q_3q_4 + q_2q_3 - q_2q_3 = p_2q_3q_4$$

$$j = 1$$

$$T = T\ q[1] = p_2q_3q_4q_1 = q_1p_2q_3q_4$$

$$H = H + T = (p_1q_2q_3q_4 + q_1q_2q_3) + q_1p_2q_3q_4$$

$$i = 2$$

$$T = F_{sys-2} - F'_{sys-2}$$

$$F_{sys-2} = p_3q_4 + q_3 \quad y \quad F'_{sys-2} = q_3$$

NOTA : F_{sys-2} y F'_{sys-2} los determinamos a partir del algoritmo Wu-Chen para sistemas CK/N:F

$$T = p_3q_4 + q_3 - q_3 = p_3q_4$$

$$j = 1$$

$$T = T\ q[1] = p_3q_4q_1 = q_1p_3q_4$$

$$j = 2$$

$$T = T\ q[2] = q_1p_3q_4q_2 = q_1q_2p_3q_4$$

$$H = H + T = (p_1q_2q_3q_4 + q_1q_2q_3) + q_1p_2q_3q_4 + q_1q_2p_3q_4$$

$$R = 1 - H = 1 - p_1q_2q_3q_4 - q_1q_2q_3 - q_1p_2q_3q_4 - q_1q_2p_3q_4$$

La confiabilidad del sistema cuyo DBC es la fig. 2.4 vendrá dada por R:

$$1 - p_1q_2q_3q_4 - q_1q_2q_3 - q_1p_2q_3q_4 - q_1q_2p_3q_4$$

2.2.6 Algoritmo Chang–Chen–Hwang para sistemas CCK/N:F ponderados

En 1998 Chang, Chen y Hwang proponen un algoritmo para evaluar la confiabilidad de sistemas consecutivos circulares (ponderados o no ponderados) a partir de expresiones de confiabilidad con menos términos que otras que se encuentran en algoritmos contemporáneos. Según los autores cuando el peso o capacidad de todos los componentes del sistema es 1, este algoritmo es más simple y eficiente que otros algoritmos de tiempo de cómputo de orden NK **[CCH]**.

Supuestos:

1) Cada componente tiene su propio w_i, p_i, q_i, para i = 1, 2, ...,N
2) El orden de los componentes se arregla de tal forma que $w_1=\max[w_i]$, i = 1, ...,N
3) Cada componente está fallado o está funcionando. Los estados de todos los componentes son mutuamente independientes.
4) Los componentes 1, 2, ..., N se arreglan en un circulo y en ese orden
5) Para K > N, $p_K = 0$ y $w_K = w_{K-N}$

Notación empleada:

i	índice del componente; i = 1, 2, …, N
w_i	peso del componente i, w_i es un entero positivo
w_{min}	w_i mínimo
w_{max}	w_i máximo; $w_{max} = w_1$ (ver supuesto # 2)
p_i, q_i	probabilidad que el componente i funcione (o falle)
$R_L(i,j)$	confiabilidad del sistema CK/N:F ponderado consistente de los componentes i, i + 1, …, j

$R_C(i,j)$ confiabilidad del sistema CCK/N:F ponderado consistente de los componentes i, i + 1, …, j

T $\max[i : \sum_{j=1}^{i-1} w_j < k,\ 1 \le i \le n+1]$

Pasos del algoritmo:

Paso 0. Definimos los vectores:

p[ ]	probabilidad de funcionamiento de los componentes
q[ ]	probabilidad de falla de los componentes
w[ ]	pesos de los componentes
Beg[ ] y End[ ]	componentes primero y último de S
Wet[ ]	peso total de S
Q[ ]	producto de probabilidades de falla
F_L[ ]	complemento de la confiabilidad

Paso 1. Introducimos y validamos los datos iniciales del sistema.

a) Introducimos N, K, w_i y q_i para i = 1, ..., N

b) Verificamos que N, K y w_i sean enteros no negativos para todo i

c) Verificamos que $0 \le q_i \le 1$ para todo i.

d) Si no se cumple alguna de las condiciones b) o c) volver al paso a)

e) Calculamos p_i a partir de la expresión $p_i = 1 - q_i$, para i =1,...,N

Paso 2. Calculamos T

T = 1

Desde i = 1 hasta i = N + 1

{ H = 0

Desde j = 1 hasta j = i – 1

{ H = H + w[j] }

Si (H < K y T < i) entonces T = i

}

Paso 3. Si T > N entonces R = 1 en caso contrario evaluamos $R_C(N,K)$:

H = 0

Desde i = 1 hasta i = T

{ $H = H + R_L(i + 1,N + i - 1) \prod_{j=1}^{i-1} q_j \; p[i]$ }

R = H

La confiabilidad $R_L(i,j)$ la calculamos a partir de la expresión $R_L(i,j) = 1 – F_L(i,j)$, y $F_L(i,j)$ lo evaluamos del algoritmo Wu – Chen para sistemas CK/N:F ponderados

Paso 4. Presentamos los resultados

La confiabilidad del sistema CCK/N:F ponderado vendrá dada por el valor de R

Ejemplo ilustrativo:

Deseamos determinar la confiabilidad del sistema cuyo DBC representamos en la Fig. 2.4. Usaremos el algoritmo Chang–Chen–Hwang para sistemas CCK/N:F ponderados.

Paso 1. Introducimos y validamos los datos iniciales del sistema.

a) Los datos de entrada son N=4, K=3, w_i=1 para todo i, q_1, q_2, q_3 y q_4

b) N, K y w_i son enteros no negativos para todo i

c) Supongamos que $0 \leq q_i \leq 1$ para todo i.

d) Calculamos p_i a partir de la expresión $p_i = 1 - q_i$, para i =1,...,N

Paso 2. Calculamos T

T = 1

i = 1

H = 0

i = 2

H = 0

j = 1

H = H + w[1] = 0 + 1

H = 1 < k = 3 y T = 1 < i = 2 $\Rightarrow$ T = i = 2

i = 3

H = 0

j = 1

H = H + w[1] = 0 + 1

j = 2

H = H + w[2] = 1 + 1

$H = 2 < k = 3$ y $T = 2 < i = 3 \Rightarrow T = i = 3$

$i = 4$

$H = 0$

$j = 1$

$H = H + w[1] = 0 + 1$

$j = 2$

$H = H + w[2] = 1 + 1$

$j = 3$

$H = H + w[3] = 2 + 1$

$i = 5$

$H = 0$

$j = 1$

$H = H + w[1] = 0 + 1$

$j = 2$

$H = H + w[2] = 1 + 1$

$j = 3$

$H = H + w[3] = 2 + 1$

$j = 4$

$H = H + w[4] = 3 + 1$

Por lo tanto $T = 3$

Paso 3. Evaluamos R_C

$H = 0$

$i = 1$

$H = H + R_L(1 + 1,4 + 2 - 1)p[1] = H + (1 - F_L(2,4))qj[1]p[1] =$

Cálculo de qj[1]:

$$qj[1] = \prod_{j=1}^{0} q_j = 1$$

$H = 0 + (1 - F_L(2,4))(1)p1 = (1 – q_2q_3q_4)p1$

$H = (1 – q_2q_3q_4)p1$

$i = 2$

$H = H + (1- F_L(2 + 1,4 + 2 - 1)qj[2]p[2] = H + (1- F_L(3,5)q_j[2]p[2]$

Cálculo de qj[1]:

$$qj[2] = \prod_{j=1}^{1} q_j = q1$$

$H = H + (1- F_L(3,5)q_1p_2 = (1 – q_2q_3q_4)p1 + (1 – q_3q_4)q_1p_2$

$H = (1 – q_2q_3q_4)p_1 + (1 – q_3q_4)q_1p_2$

$i = 3$

$H = H + (1 - F_L(3 + 1,4 + 3 - 1)qj[3]p[3] = H + (1 - F_L(4,6)qj[3]p[3]$

Cálculo de qj[3]:

$$qj[3] = \prod_{j=1}^{2} q_j = q_1q_2$$

$$H = H + (1 - F_L(4,6)q_1q_2p_3 = H + (1 - q_4)q_1q_2p_3$$

$$H = (1 - q_2q_3q_4)p1 + (1 - q_3q_4)q_1p_2 + (1 - q_4)q_1q_2p_3$$

Finalmente R = H

Paso 4. Presentamos los resultados

La confiabilidad del sistema cuyo DBC es la Fig. 11 vendrá dada por el valor de R

$$(1 - q_2q_3q_4)p_1 + (1 - q_3q_4)q_1p_2 + (1 - q_4)q_1q_2p_3$$

Esta expresión es equivalente a la que se obtiene con el algoritmo Wu-Chen para sistemas CCK/N:F no ponderados, debido a que las capacidades de los elementos son iguales.

2.2.7 Algoritmo Zuo para sistemas LCCS con componentes biestado

En 1993 Zuo desarrolla un algoritmo de tiempo de cómputo de orden NK para la evaluación de confiabilidad de sistemas LCCS donde K (K < N) es la capacidad máxima de transmisión de todos los componentes **[ZUO]**.

Supuestos:

a) Cada componente y el sistema pueden estar sólo en uno de los dos estados siguientes: funcionando o fallado

b) Los estados de todos los componentes son mutuamente independientes

c) La fuente y la carga son totalmente confiables

Notación empleada:

N número de componentes del sistema

k_i capacidad de transmisión del componente i, o el número máximo de componentes inmediatamente sucesivos que el componente i puede alcanzar directamente

p_i, q_i probabilidad que el componente i funcione (falle)

R(i,j) confiabilidad de un sistema LCCS con fuente i, sumidero j, y componentes i + 1, i + 2, ..., j – 1, dado que el componente l tiene una capacidad de transmisión k_l ($0 \le i \le l \le j - 1 \le N$)

Pasos del algoritmo:

Paso 0. Definimos los vectores:

p[ ] probabilidad de funcionamiento de los componentes

k[ ] capacidad de transmisión

R[ ] confiabilidad

Paso 1. Introducimos y validamos los datos iniciales del sistema.

a) Introducimos N, k_i y p_i con $0 \le i \le N-1$

b) Verificamos que N y k_i sean enteros no negativos para todo i

c) Verificamos que $0 \le p_i \le 1$ para todo i.

d) Si no se cumple alguna de las condiciones b) o c) volver al paso a)

Paso 2. Evaluación del vector de confiabilidad R

Desde i = N – 2 hasta i = 0

{ Si i + k[i] ≥ N – 1

entonces R[i] = 1

```
        en caso contrario:
        {       R[i] = 0
                TEMP = 1
                Desde j = i hasta j = i + k[i]
                Si j + k[j] > i + k[i]
                {       R[i] = R[i] + TEMP p[i] R[j]
                        TEMP = TEMP (1 - p[j])
                }
        }
}
```

Paso 3. Presentamos los resultados

En el elemento R[0] del vector de confiabilidad R encontramos la confiabilidad del sistema LCCS.

Ejemplo ilustrativo:

Deseamos determinar la confiabilidad del sistema cuyo DBC representamos en la Fig. 2.5. Usaremos el algoritmo Zuo para sistemas LCCS con componentes biestado.

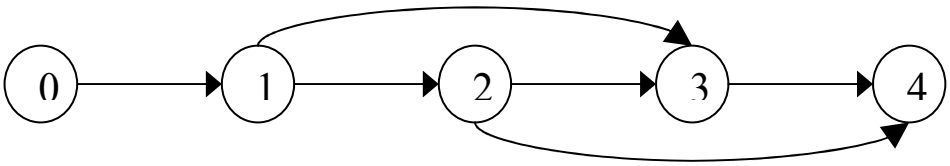


Fig. 2.5 Sistema LCCS de 3 componentes

Paso 1. Introducimos y validamos los datos iniciales del sistema.

a) Los datos de entrada son N=5 (incluyendo fuente y carga), $k_0 = 1$, $k_1 = 2$, $k_2 = 2$, $k_3 = 1$, p_1, p_2, p_3 y p_4.

b) N y k_i son enteros no negativos para todo i

c) Supongamos que $0 \leq p_i \leq 1$ para todo i.

Paso 2. $i = N - 2 = 3$

$3 + k_3 = 3 + 1 = 4$ y $N - 1 = 4 \Rightarrow R(3) = 1$

$i = 2$

$2 + k_2 = 2 + 2 = 4$ y $N - 1 = 4 \Rightarrow R(2) = 1$

$i = 1$

$1 + k_1 = 1 + 2 = 3$ y $n - 1 = 4 \Rightarrow$

$R(1) = 0$

$temp = 1$

$j = 2$

$2 + k_2 = 2 + 2 = 4$ y $1 + k_1 = 3 \Rightarrow$

$R(1) = R(1) + temp\ p_2\ R(2) = 0 + (1)\ p_2\ (1)$

$R(1) = p_2$

$temp = temp\ (1 - p_2) = (1)\ (1 - p_2)$

$temp = q_2$

$j = 3$

$3 + k_3 = 4$ y $1 + k_1 = 3 \Rightarrow$

$R(1) = R(1) + temp\ p_3\ R(3) = p_2 + q_2\ p_3\ (1)$

$R(1) = p_2 + q_2 p_3$

$i = 0$

$0 + k_0 = 1 + 1 = 2$ y $n - 1 = 4 \Rightarrow$

$R(0) = 0$

$temp = 1$

$j = 1$

$1 + k_1 = 3$ y $0 + k_0 = 1 \Rightarrow$

$R(0) = R(0) + temp\ p_1\ R(1) = 0 + (1)\ p_1\ (p_2 + q_2\ p_3)$

$R(0) = p_1\ (p_2 + q_2 p_3)$

La confiabilidad del sistema cuyo DBC es la Fig. 2.5 vendrá dada por R(0):

$$p_1\ (p_2 + q_2 p_3)$$

2.2.8 Algoritmo Kosow – Preuss para sistemas LCCS con componentes multiestado

En 1995 Kosow y Preuss desarrollaron un algoritmo para evaluar la confiabilidad de sistemas LCCMS, sistemas que no maneja el algoritmo Zuo ya que éste supone que los componentes del sistema son biestado.

Supuestos:

a) El sistema tiene C_i componentes ordenados consecutivamente con $i \in [0,N]$, y el sumidero $C_{n+1.}$ C_0 es la fuente.

b) El estado Z_i de C_i es una variable aleatoria discreta con una función de distribución de probabilidad:

$$\Pr\{Z_i = j\} = p_{ij}\ ,\ \sum_j p_{ij} = 1$$

$Z_i = j$ implica una ruta desde C_i hasta cada uno de los $C_{i+1}, C_{i+2}, ..., C_{i+\min(j,N-i+1)}$

$Z_i = 0$ implica el estado de falla de C_i.

c) Los Z_i son mutuamente independientes.

d) El sumidero C_{N+1} es absolutamente confiable.

e) 5) El sistema esta bajo falla si y solo si no hay camino desde C_0 hasta C_{N+1}

Notación empleada:

i	índice del componente C_i, $i \in [0,N]$
N	número de componentes del sistema (excluyendo fuente y carga)
C_i, C_{N+1}	componente i, carga o sumidero
k_i	número de amplificadores del componente C_i
j	índice que representa el estado del componente, $j \in [0, k_i]$

$p_{i,j}$ probabilidad que C_i se encuentre en el estado j

Pasos del algoritmo:

Paso 1. Introducimos y validamos los datos iniciales del sistema.

a) Introducimos N, k_i con $0 \le i \le N$ y $p_{i,j}$ con $0 \le i \le N$, $0 \le j \le k_i$

b) Verificamos que N y k_i sean enteros no negativos para todo i

c) Verificamos que $0 \le p_{i,j} \le 1$ para todo i y j.

d) Si no se cumple alguna de las condiciones b) o c) volver al paso a)

Paso 2. Si $N = 0 \Rightarrow R(0) = 1 - p_{0,0}$ en caso contrario pasamos a evaluar el vector de confiabilidad R

Paso 3. $K = \text{máximo}(k_i)$ $0 \le i \le N$

$R(0) = 1 - p_{0,0}$

$j = 1$

Desde $j = 1$ hasta $j = k_0$

{ $R(j) = 0$

Desde $v = 1$ hasta $v = j$

{ $R(j) = R(j) + A(v, j)\, R(j - v)$ }

$R(j) = R(j) + A(j+1,j)$

}

Desde $j = k_0$ hasta $j = N$

{ $R(j) = 0$

Desde v = 1 hasta v = mínimo(K,N)

{ R(j) = R(j) + A(v,j) R(j – v) }

}

Cálculo de los A(x,y) :

{ Si x = 1 $\Rightarrow$ A(x,y) = 1 – $p_{y,0}$

Si x = 2 $\Rightarrow$

{ H = 0

Desde j = 2 hasta j = k_{y-1}

{ H = H + $p_{y-1, j}$ }

A(x,y) = $p_{y,0}$ H

Si x = 3 $\Rightarrow$

{ P = 1

Desde r = 1 hasta r = x - 2

{ H = 0

Desde s = 0 hasta s = mínimo(r,k_{y-r})

{ H = H + $p_{y-r,s}$ }

P = P H

}

H = 0

Desde j = x hasta j = k_{y-x+1}

{ $H = H + p_{y-x+1, j}$ }

$A(x,y) = p_{y,0}\ P\ H$

}

Paso 4. Presentamos los resultados:

La confiabilidad del sistema vendrá dada por R(N)

Ejemplo Ilustrativo:

Deseamos determinar la confiabilidad del sistema cuyo DBC representamos en la Fig. 2.5. Usaremos el algoritmo Kosow - Preuss para sistemas LCCS con componentes multiestado.

Paso 1. Introducimos y validamos los datos iniciales del sistema.

a) Los datos de entrada son N=3 (no se incluyen fuente ni carga),

$k_0 = 1$; $k_1 = 2$; $k_2 = 2$; $k_3 = 1$;

$p_{0,0}$; $p_{0,1}$; $p_{1,0}$; $p_{1,1}$; $p_{1,2}$; $p_{2,0}$; $p_{2,1}$; $p_{2,2}$; $p_{3,0}$; $p_{3,1}$

b) N y k_i son enteros no negativos para todo i

c) Suponemos que $0 \leq p_{i,j} \leq 1$ para todo i y j.

Paso 2. $N \neq 0 \Rightarrow$

Paso 3. Evaluamos el vector de confiabilidad R

$k = \text{máximo}\ (k_i) = 2$

$R(0) = 1 - p_{0,0}$

$j = 1$

$R(1) = 0$

v = 1

$R(1) = R(1) + A(1,1)\, R(0)$

Calculo de A(1,1):

Si v = 1 $\Rightarrow A(1,1) = 1 - p_{1,0}$

$\Rightarrow R(1) = 0 + (1 - p_{1,0})\,(1 - p_{0,0})$

v = 2

$R(1) = R(1) + A(2,1)R(-1)$ siendo $R(-1) = 0$

$\Rightarrow R(1) = (1 - p_{1,0})(1 - p_{0,0})$ } }

j = 2

$R(2) = 0$

v = 1

{ $R(2) = R(2) + A(1,2)R(1)$

Cálculo de A(1,2):

Si v = 1 $\Rightarrow A(1,2) = 1 - p_{2,0}$

$\Rightarrow R(2) = 0 + (1 - p_{2,0})(1 - p_{1,0})(1 - p_{0,0})$

$R(2) = (1 - p_{2,0})(1 - p_{1,0})(1 - p_{0,0})$ }

v = 2

{ $R(2) = R(2) + A(2,2)R(0)$

Cálculo de A(2,2):

Si v = 2

H = 0

j = 2

{ H = H + p1,2

H = $p_{1,2}$

$\Rightarrow$ A(2,2) = $p_{2,0}$ $p_{1,2}$ }

R(2) = (1–$p_{2,0}$) (1–$p_{1,0}$) (1–$p_{0,0}$) + $p_{2,0}$ $p_{1,2}$ (1-$p_{0,0}$) }

j = 3

{ R(3) = 0

v = 1

R(3) = R(3) + A(1,3) R(2)

Cálculo de A(1,3):

Si v = 1 $\Rightarrow$ A(1,3) = 1 – $p_{3,0}$

R(3) = 0 + (1 – $p_{3,0}$) R(2)

v = 2

R(3) = R(3) + A(2,3) R(1)

Cálculo de A(2,3) :

Si v = 2 $\Rightarrow$

{ H = 0

j = 2

H = H + p2,2

$\Rightarrow A(2,3) = p_{3,0}\, p_{2,2}$ }

$R(3) = (1 - p_{3,0})R(2) + p_{3,0}\, p_{2,2}\, R(1)$, si sustituimos R(1) y R(2) $\Rightarrow$

$R(3) = (1 - p_{3,0})\,[(1-p_{2,0})\,(1-p_{1,0})\,(1-p_{0,0}) + p_{2,0}\, p_{1,2}\,(1-p_{0,0})] +$

$+ p_{3,0}\, p_{2,2}\,(1 - p_{1,0})(1 - p_{0,0})$

Paso 4. Presentamos los resultados:

La confiabilidad del sistema cuyo DBC es la Fig. 2.5 vendrá dada por R(3).

Si efectuamos las siguientes sustituciones:

$p_{0,0} = 0$ $\quad p_{0,1} = 1$ $\quad p_{0,2} = 0$

$p_{1,0} = q_1$ $\quad p_{1,1} = 0$ $\quad p_{1,2} = p_1$

$p_{2,0} = q_2$ $\quad p_{2,1} = 0$ $\quad p_{2,2} = p_2$

$p_{3,0} = q_3$ $\quad p_{3,1} = p_3$ $\quad p_{3,2} = 0,$

la expresión R(3) es idéntica a la expresión obtenida con el algoritmo Zuo para sistemas LCCS con componentes biestado.

En el siguiente capítulo presentamos un análisis comparativo de las características cuantitativas de los algoritmos de evaluación de confiabilidad que estudiamos en este capítulo y efectuamos un análisis cualitativo de las funciones de confiabilidad y costo de los sistemas que definimos en el capítulo I.

CAPÍTULO III

COMPARACIÓN Y ANÁLISIS DE RESULTADOS

3.1 PROGRAMAS UTILIZADOS

Los algoritmos presentados en el capítulo 2 fueron programados en BORLAND C++ Versión 3.1. Para efectos del análisis de este capítulo en la Tabla 3.1 mostramos el nombre que le asignamos a cada programa:

Tabla 3.1 Clasificación de los programas

Nombre del algoritmo	**Sistema sobre el cual se aplica**	**Nombre asignado al programa**
Wu – Chen para sistemas K/N:G no ponderados	K/N:G sin restricciones de capacidad	WCH01
Wu – Chen para sistemas K/N:G ponderados	K/N:G con restricciones de capacidad	WCH02
Kosow – Preuss para sistemas CK/N:F no ponderados	CK/N:F sin restricciones de capacidad	KP01
Wu – Chen para sistemas CK/N:F ponderados	CK/N:F con restricciones de capacidad	WCH03
Wu – Chen para sistemas CCK/N:F no ponderados	CCK/N:F sin restricciones de capacidad	WCH04
Chang - Chen – Hwang para sistemas CCK/N:F ponderados	CCK/N:F con restricciones de capacidad	CCH01
Zuo para sistemas LCCS con componentes biestado	LCCS con componentes biestado	ZT01
Kosow – Preuss para sistemas LCCS con componentes multiestado	LCCMS con componentes multiestado	KP02

Por ejemplo, cuando hagamos referencia al programa WCH02 estaremos hablando del programa del algoritmo Wu - Chen para sistemas K/N:G ponderados. Los programas solicitan los datos que necesitan para los cálculos (N, K, W, etc.) y devuelven el valor numérico (no simbólico) de la confiabilidad del sistema.

Los programas de la Tabla 3.1 fueron revisados y depurados con ayuda de gran cantidad de ejemplos. Los programas fuentes se presentan en el **Apéndice A**.

3.2 ANÁLISIS COMPARATIVO

3.2.1 Tiempos de ejecución.

En la comparación de algoritmos de evaluación de confiabilidad de sistemas complejos, el tiempo de ejecución constituye una poderosa herramienta de análisis (**[YD]** y **[M]**). Con relación a este aspecto, en los algoritmos de la tabla 3.1 encontramos las siguientes particularidades:

a) Cuando los algoritmos trabajan con SPR no ponderados los tiempos de ejecución dependen linealmente del número de componentes del sistema "N" y del factor de redundancia parcial "K". Por ejemplo, supongamos que el programa WCH01 para sistemas K/N:G no ponderados evalúa la confiabilidad de un sistema 1/10:G en 1 ms, entonces podemos decir con toda seguridad que este programa para un sistema 1/100:G tiene un tiempo de ejecución no mayor de 10 ms.

b) Los algoritmos -en cada tipo particular de sistema- presentan muy pocas diferencias en cuanto a estructura interna y a necesidad de recursos de tiempo y memoria.

Por lo expuesto consideramos que no es necesario efectuar un análisis cuantitativo de tiempos de ejecución de los algoritmos de la Tabla 3.1.

3.2.2 Funciones de Confiabilidad y Costo.

El siguiente parámetro de comparación tiene que ver con la relación que existe entre confiabilidad y costo en un sistema dado, relación que puede analizarse a partir del estudio conjunto de las funciones de confiabilidad y costo.

El método que emplearemos para efectuar este análisis será el siguiente:

1) Definimos un sistema ejemplo para los sistemas K/N:G, CK/N:F, CCK/N:F, LCCS y LCCMS,

2) Determinamos la función de confiabilidad de los sistemas para distintos valores de K y W, utilizamos los algoritmos de evaluación de confiabilidad para construir cada función de confiabilidad.

3) Graficamos las funciones de costo de los sistemas vs. K y W.

4) Finalmente, efectuamos un análisis cualitativo de la relación entre Confiabilidad y Costo para los sistemas presentados.

3.2.2.1 Sistemas K/N:G

Para estos sistemas se definen dos casos:

Caso a) Sistemas no ponderados,

Caso b) Sistemas ponderados.

En el caso "a" los datos de entrada son N = 10, $W_i = 1$ con $1 \leq i \leq N$ y variamos K en el rango [0, 10]. En el caso "b" los datos de entrada son N = 10, variamos K en el rango [0, 10] y presentamos tres subcasos:

Subcaso b-1) $W_1 = 2$, $W_i = 1$ con $2 \leq i \leq 10$,

Subcaso b-2) $W_i = 2$ con $1 \leq i \leq 5$ y $W_j = 1$ con $6 \leq j \leq 10$, y

Subcaso b-3) $W_i = 2$ con $1 \leq i \leq 10$.

Suponemos que $p_i = 0{,}5$ con $1 \leq i \leq 10$ para todos los casos. Utilizaremos el programa WCH01 para el caso "a" y el programa WCH02 para el caso "b". En la Fig. 3.1 presentamos la función de confiabilidad del sistema K/N:G con los casos y subcasos definidos arriba:

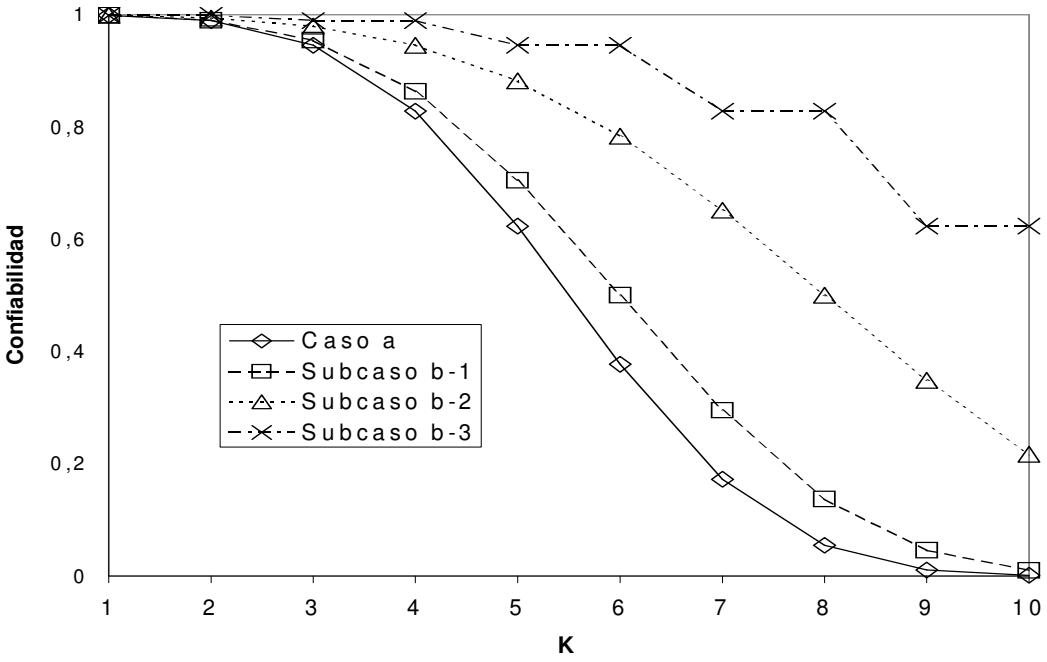


Fig. 3.1 Función de Confiabilidad de Sistemas K/N:G

En la Fig. 3.1 podemos observar para los casos "a", "b-1", "b-2" y "b-3" que la confiabilidad del sistema disminuye si aumenta K. Esto comportamiento se debe a que cuando K aumenta y tiende a N el sistema se hace menos confiable ya que para que el sistema funcione necesita que mayor cantidad de componentes funcionen simultáneamente, es decir el sistema tiende a comportarse como un sistema serie.

Por el contrario, si K disminuye y tiende a 1 la confiabilidad del sistema para los casos "a", "b-1", "b-2" y "b-3" aumenta. Esto se debe a que cuando K disminuye el sistema se hace más confiable ya que para que funcione necesita que menos cantidad de componentes funcionen simultáneamente, es decir el sistema tiende a comportarse como un sistema paralelo.

En los sistemas K/N:G en cuanto a la confiabilidad se cumple la relación caso "b-3" > caso "b-2" > caso "b-1" > caso "a" ya que en la medida que aumentamos la capacidad de transmisión de los componentes el sistema se torna más confiable ya que si falla un componente habrá otro que ayude o compense al componente fallado y permitirá que el sistema funcione.

Para construir la función de costo de los casos "a" y "b", utilizaremos el modelo de la expresión (1.2):

$$C = NC_1\, g\,(W) / K + r\, C_2\, (1 - R) \qquad (1.2)$$

Donde N = 10, variamos K en el rango [0, 10], obtenemos R de los programas WCH01 (caso "a") y WCH02 (caso "b") y suponemos que $C_1 = 10$, $C_2 = 1000$ y $r = 0{,}98$. La función g(W) dependerá del subcaso que estemos estudiando:

Subcaso b-1) Si $W_1 = 2$, $W_i = 1$ con $2 \le i \le 10 \Rightarrow g(W) = 1.10$

Subcaso b-2) Si $W_i = 2$ con $1 \le i \le 5$ y $W_j = 1$ con $6 \le j \le 10 \Rightarrow g(W) = 1.5$

Subcaso b-3) Si $W_i = 2$ con $1 \le i \le 10 \Rightarrow g(W) = 2$

En la Fig. 3.2 de la página siguiente presentamos la función de costo del sistema K/N:G con los casos y subcasos definidos arriba:

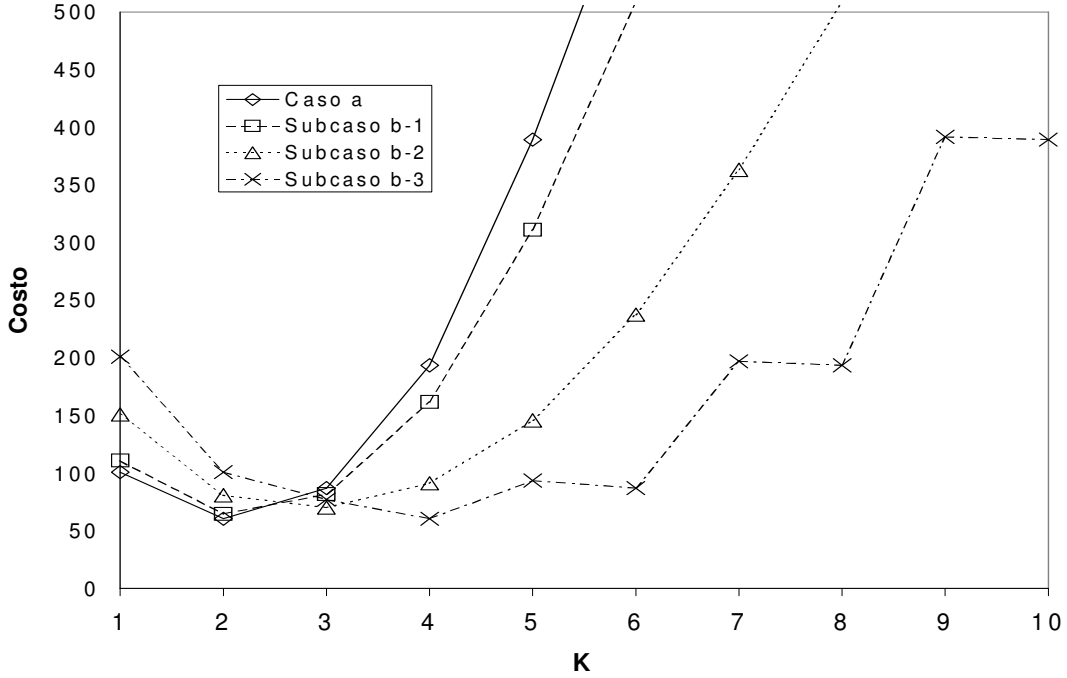


Fig. 3.2 Función de Costo de Sistemas K/N:G

Nos interesan las zonas de las gráficas donde las funciones de confiabilidad y costo presentan sus máximos y mínimos, respectivamente. En el sistema K/N:G que tomamos como ejemplo, las funciones de confiabilidad y costo presentan sus máximos y mínimos para valores de K comprendidos en [1,4]. Si K > 5 disminuye la confiabilidad y se eleva el costo (véanse las Fig. 3.1 y 3.2).

3.2.2.2 Sistemas CK/N:F

Para estos sistemas se definen dos casos:

Caso a) Sistemas no ponderados,

Caso b) Sistemas ponderados.

En el caso "a" los datos de entrada son N = 10, $W_i = 1$ con $1 \leq i \leq N$ y variamos K en el rango [0, 10]. En el caso "b" los datos de entrada son N = 10, variamos K en el rango [0,10] y presentamos cuatro subcasos:

Subcaso b-1) $W_1 = 2$, $W_i = 1$ con $2 \leq i \leq 10$

Subcaso b-2) $W_i = 2$ con $1 \leq i \leq 5$ y $W_j = 1$ con $6 \leq j \leq 10$

Subcaso b-3) $W_i = 2$ con $1 \leq i \leq 10$.

Subcaso b-4) $W_i = 2$ con i = 1, 3, 5 , 7 y 10, $W_j = 1$ con j = 2, 4, 6, 8 y 9

Suponemos que $p_i = 0{,}5$ con $1 \leq i \leq 10$ para todos los casos. Utilizaremos el programa KP01 para el caso "a" y el programa WCH03 para el caso "b". En la Fig. 3.3 presentamos la función de confiabilidad del sistema CK/N:F con los casos y subcasos definidos arriba:

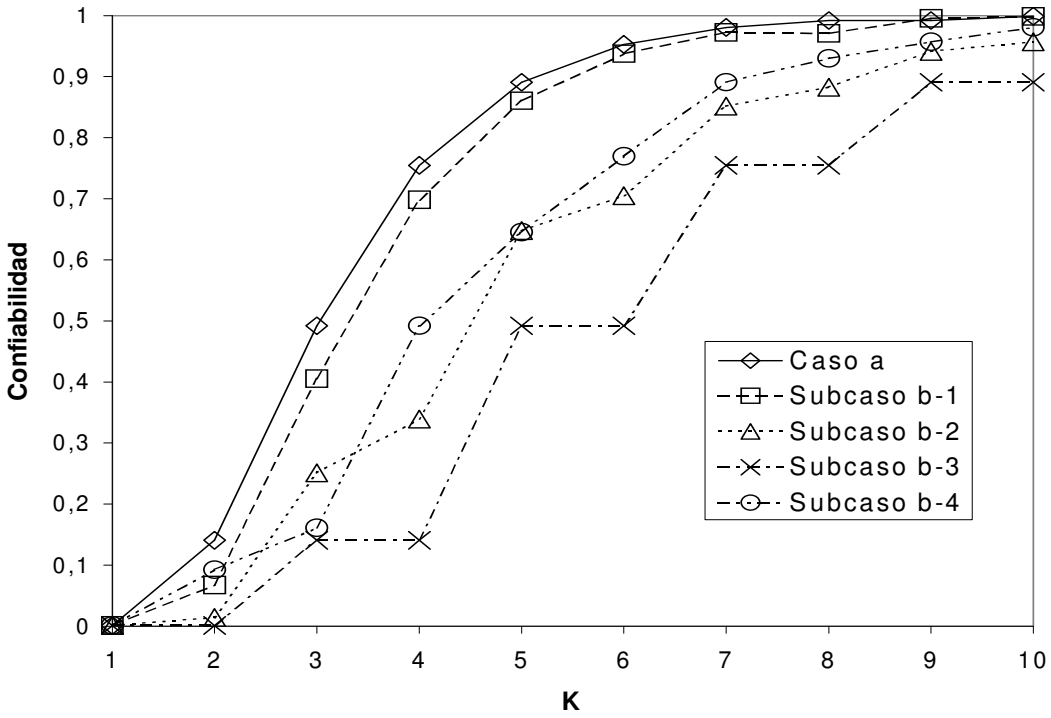


Fig. 3.3 Función de Confiabilidad de Sistemas CK/N:F

Utilizaremos el modelo de la expresión (1.3) para construir la función de costo de los casos "a" y "b":

$$C = NC_1\, g\,(W)\, K + r\, C_2\,(1 - R) \qquad (1.3)$$

Donde N = 10, variamos K en el rango [0,10]. En los sistemas CK/N:F obtenemos R a partir de los programas KP01 (caso "a") y WCH03 (caso "b"). Suponemos que C_1 = 10, C_2 = 400 y r = 0,98. La función g(W) dependerá del subcaso que estemos estudiando:

Subcaso b-1) Si $W_1 = 2$, $W_i = 1$ con $2 \leq i \leq 10 \Rightarrow g(W) = 1.10$

Subcaso b-2) Si $W_i = 2$ con $1 \leq i \leq 5$ y $W_j = 1$ con $6 \leq j \leq 10 \Rightarrow g(W) = 1.5$

Subcaso b-3) Si $W_i = 2$ con $1 \leq i \leq 10 \Rightarrow g(W) = 2$

Subcaso b-4) Si $W_i = 2$ con i = 1, 3, 5, 7 y 10, $W_j = 1$ con j = 2, 4, 6, 8 y 9 $\Rightarrow g(W) = 1.5$

En la Fig. 3.4 presentamos la función de costo del sistema CK/N:F con los casos y subcasos definidos arriba:

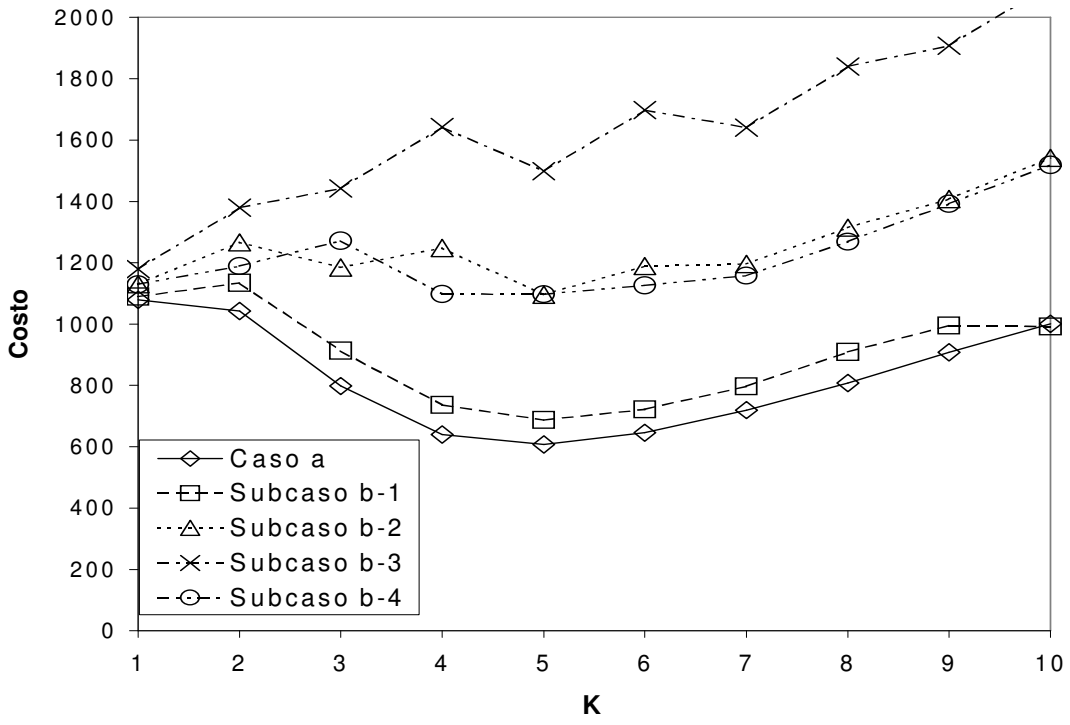


Fig. 3.4 Función de Costo de Sistemas CK/N:F

3.2.2.3 Sistemas CCK/N:F

Para estos sistemas se definen dos casos:

Caso a) Sistemas no ponderados,

Caso b) Sistemas ponderados.

En el caso "a" los datos de entrada son N = 10, $W_i = 1$ con $1 \leq i \leq N$ y variamos K en el rango [0, 10]. En el caso "b" los datos de entrada son N = 10, variamos K en el rango [0,10] y presentamos cuatro subcasos:

Subcaso b-1) $W_1 = 2$, $W_i = 1$ con $2 \leq i \leq 10$

Subcaso b-2) $W_i = 2$ con $1 \leq i \leq 5$ y $W_j = 1$ con $6 \leq j \leq 10$

Subcaso b-3) $W_i = 2$ con $1 \leq i \leq 10$.

Subcaso b-4) $W_i = 2$ con i = 1, 3, 5 , 7 y 10, $W_j = 1$ con j = 2, 4, 6, 8 y 9

Suponemos que $p_i = 0{,}5$ con $1 \leq i \leq 10$ para todos los casos. En los sistemas CCK/N:F utilizaremos el programa WCH04 para el caso "a" y el programa CCH01 para el caso "b". En la Fig. 3.5 de la página siguiente presentamos la función de confiabilidad del sistema CCK/N:F con los casos y subcasos definidos arriba:

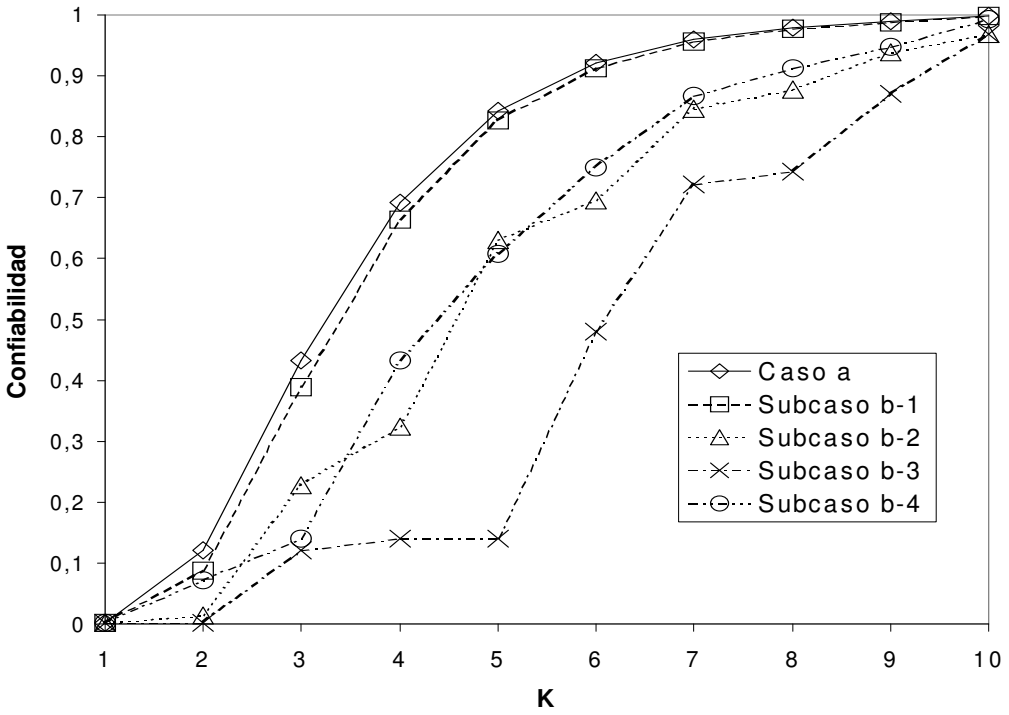


Fig. 3.5 Función de Confiabilidad de Sistemas CCK/N:F

En los sistemas CK/N:F y CCK/N:F la confiabilidad para los casos “a”, “b-1”, “b-2”, “b-3” y “b-4” aumenta si K aumenta. Esto se debe a que cuando K aumenta y tiende a N el sistema se hace más confiable ya que para que funcione necesita que menos cantidad de componentes consecutivos funcionen simultáneamente, el sistema tiende a comportarse como un sistema paralelo.

Si por el contrario K disminuye y tiende a 1 la confiabilidad del sistema para los casos “a”, “b-1”, “b-2”, “b-3” y “b-4” disminuye. Este comportamiento se debe a que cuando K disminuye el sistema se hace menos confiable ya que para que funcione necesita que mayor cantidad de componentes consecutivos funcionen simultáneamente, el sistema tiende a comportarse como un sistema serie.

En los sistemas CK/N:F y CCK/N:F en cuanto a confiabilidad no se cumple la relación caso “b-4” > caso “b-3” > caso “b-2” > caso “b-1” > caso “a” ya que se presenta un factor que no esta presente en los sistemas K/N:G, la posición relativa de los componentes. En las Fig. 3.3 y 3.5 el subcaso “b-4” presenta una confiabilidad mayor –

solo para algunos valores de K- que el subcaso "b-2" y la única diferencia entre ambos subcasos es la posición relativa de los componentes que tienen doble capacidad de transmisión, es decir la reubicación o relocalización de componentes en sistemas CK/N:F y CCK/N:F puede aumentar o disminuir la confiabilidad del sistema.

Para construir la función de costo de los casos "a" y "b", utilizaremos el modelo de la expresión (1.3):

$$C = NC_1\, g(W)\, K + r\, C_2\, (1 - R) \qquad (1.3)$$

Donde N = 10, variamos K en el rango [0,10]. En los sistemas CCK/N:F obtenemos R a partir de WCH04 (caso "a") y CCH01 (caso "b"). Suponemos que $C_1 = 10$, $C_2 = 400$ y $r = 0{,}98$. La función g(W) dependerá del subcaso que estemos estudiando:

Subcaso b-1) Si $W_1 = 2$, $W_i = 1$ con $2 \le i \le 10 \Rightarrow g(W) = 1.10$

Subcaso b-2) Si $W_i = 2$ con $1 \le i \le 5$ y $W_j = 1$ con $6 \le j \le 10 \Rightarrow g(W) = 1.5$

Subcaso b-3) Si $W_i = 2$ con $1 \le i \le 10 \Rightarrow g(W) = 2$

Subcaso b-4) Si $W_i = 2$ con i = 1, 3, 5, 7 y 10, $W_j = 1$ con j = 2, 4, 6, 8 y 9
$\Rightarrow g(W) = 1.5$

En la Fig. 3.6 de la página siguiente presentamos la función de costo del sistema CCK/N:F con los casos y subcasos definidos arriba:

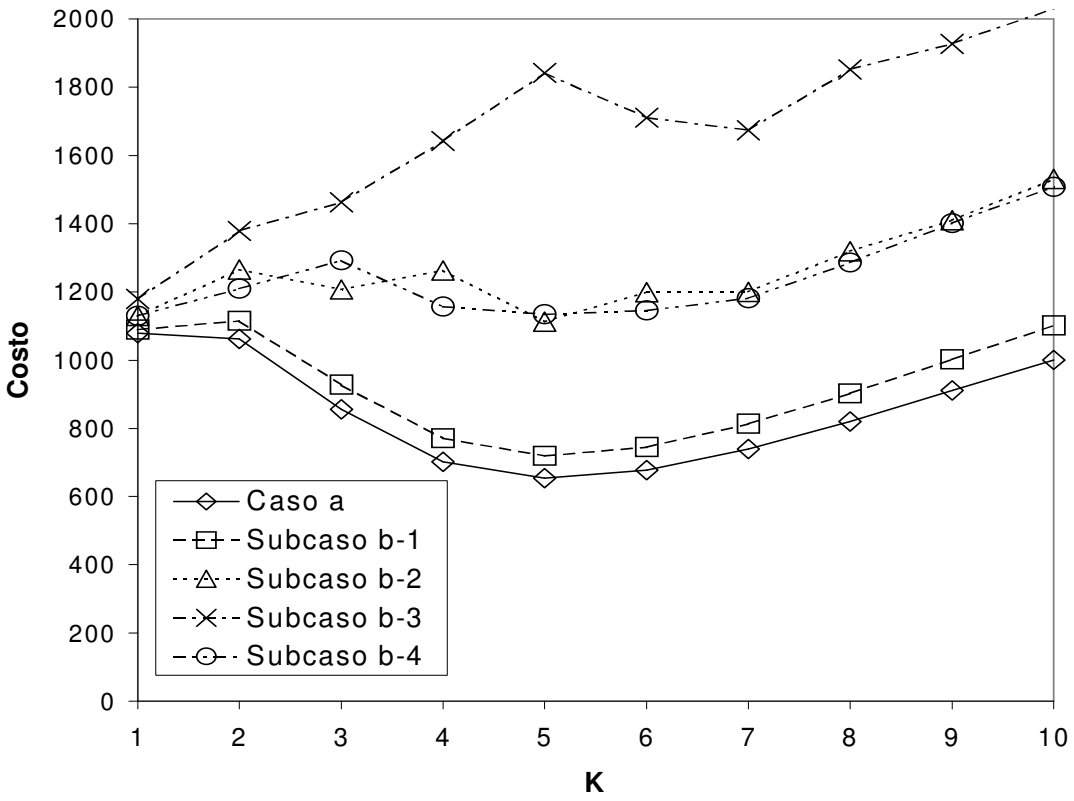


Fig. 3.6 Función de Costo de Sistemas CCK/N:F

En los sistemas CK/N:F y CCK/N:F que tomamos como ejemplo, el máximo de la función de confiabilidad y el mínimo de la función de costo se presentan para valores de $K \in [3,7]$, para otros valores de K la confiabilidad aumenta o disminuye pero el costo se eleva (véanse las Fig. 3.3, 3.4, 3.5 y 3.6).

3.2.2.4 Sistemas LCCS

Para estos sistemas se definen dos casos:

Caso a) El componente i solo posee una probabilidad de funcionamiento p_i, utilizamos el programa ZT01 para este caso, y

Caso b) El componente i tiene k_i probabilidades de funcionamiento, utilizamos el programa KP02 para este caso.

Los datos de entrada para ambos casos son:

1) N = 10

2) Variamos k_i en el rango [0,10], si k_i = 5, significa que tendremos un sistema donde $k_0=k_1=k_2=k_3=k_4=k_5=k_6=5$ y $k_7=k_8=k_9=k_{10}=1$,

3) Para el caso "a" suponemos que $p_i = 0.5$ con $0 \leq i \leq 10$

4) Para el caso "b" suponemos que $p_{0,j} = 0.5$ para todo j, y $p_{i,j} = 0.5/k_i$ para todo j e $i \neq 0$

En la Fig. 3.7 presentamos la función de confiabilidad del sistema LCCS con los casos y subcasos definidos arriba:

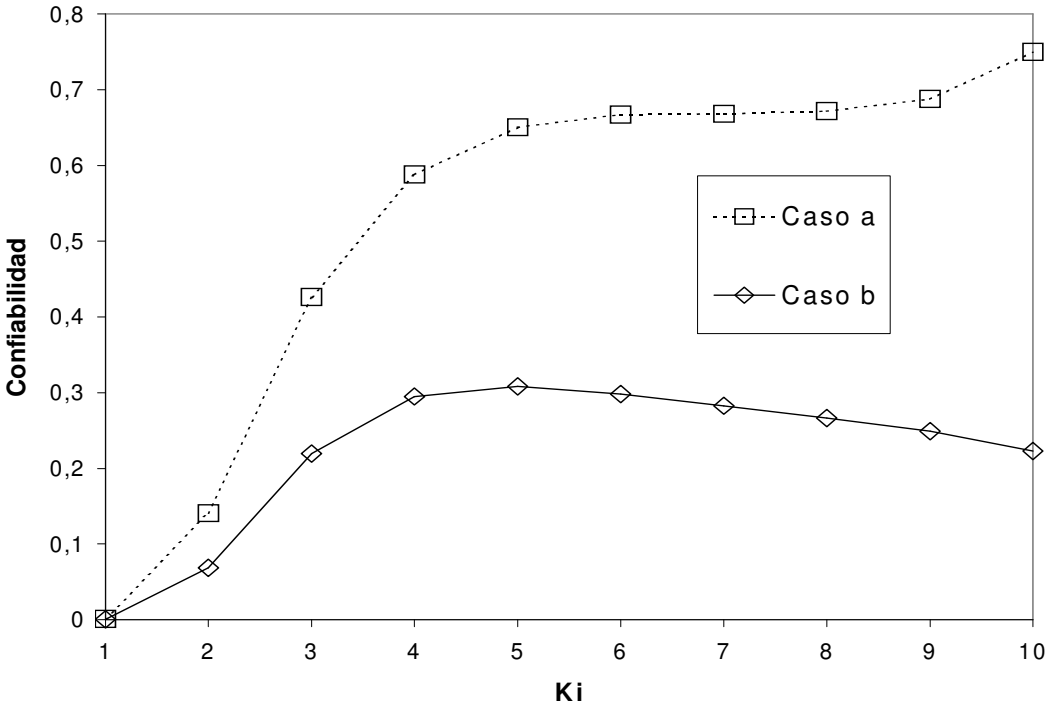


Fig. 3.7 Función de Confiabilidad de Sistemas LCCS

En los sistemas LCCS la confiabilidad del caso "a" siempre es mayor que la confiabilidad del caso "b" ya que en el caso "b" se disminuye la probabilidad de funcionamiento de los componentes, en la práctica es como si se redujera la potencia y el alcance de transmisión de los elementos transmisores del sistema. La función de

confiabilidad en general aumenta con aumentos de k_i pero llega un punto donde para incrementos de k_i la confiabilidad aumenta o disminuye muy poco (véase la Fig. 3.7).

Para construir la función de costo para los casos “a” y “b”, utilizaremos el modelo de la expresión (1.4):

$$C = NC_1\, g\,(k_i) + r\, C_2\,(1 - R) \qquad (1.4)$$

Los parámetros del modelo son N = 10, C1 = 10, r = 0.98, C2 = 1000. El valor de R lo calculamos a partir de ZT01 (caso “a”) o KP02 (caso “b”) y g(ki) vendrá dada por la expresión (3.1):

$$g(k_i) = ((N+1) + \Sigma k) / (N+1) \qquad (3.1)$$

donde

$$\Sigma k = (k_o-1) + (k_1-1) + \ldots + (k_N-1)$$

En la Fig. 3.8 de la página siguiente presentamos la función de costo del sistema LCCS con los casos y subcasos definidos arriba:

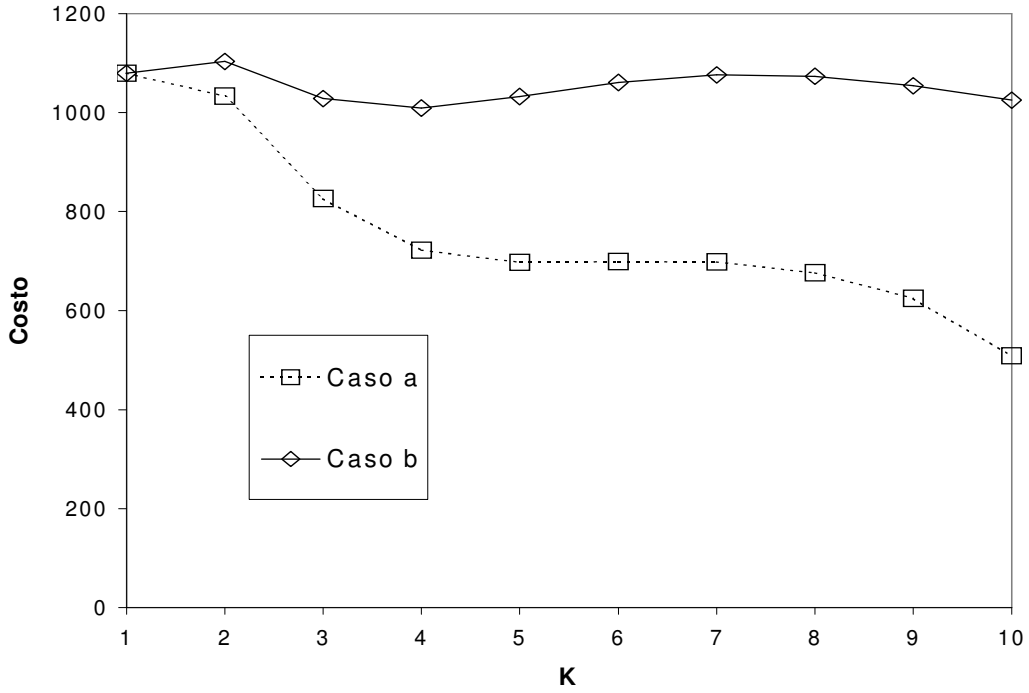


Fig. 3.8 Función de Costo de Sistemas LCCS

En el sistema LCCS tomado como ejemplo, la función de confiabilidad aumenta y la función de costo disminuye hacia el extremo derecho de las gráficas, donde K_i es cercano a 10 (véanse las Fig. 3.7 y 3.8).

3.3 Aplicaciones del análisis comparativo

En general, el análisis efectuado sobre las funciones de confiabilidad y costo de los SPR estudiados nos permite:

1. *Realizar análisis de sensibilidad de sistemas existentes o en fases de diseño:*

La confiabilidad y el costo de los sistemas se ve afectada por cambios en la configuración (adición o eliminación de componentes), por modificaciones del factor de redundancia parcial (aumento o disminución de K) y por variaciones en la capacidad de transmisión de los elementos (aumento o disminución de W). El análisis de las funciones

de confiabilidad y costo permite estimar la magnitud de estos cambios, orientándonos durante el proceso de diseño y operación de los sistemas.

2. *Diseñar SPR que presenten confiabilidad y costo óptimos:*

Las técnicas de optimización no lineales y el análisis conjunto de las funciones de confiabilidad y costo de los SPR nos permite seleccionar el sistema que presente confiabilidad y costo óptimos. El proceso de optimización está sujeto a consideraciones de confiabilidad, costo y estructura.

3. *Efectuar estudios comparativos entre distintos SPR:*

Es posible que en las primeras fases de diseño de un sistema tengamos que escoger de entre un abanico de SPR, por ejemplo a nivel de comportamiento la función de confiabilidad de los sistemas CK/N:F y CCK/N:F son muy similares pero a nivel de costo un sistema CCK/N:F puede resultar más costoso que un sistema CK/N:F. El estudiar el comportamiento de esos sistemas a variaciones en los distintos parámetros nos ayudaría a tomar una mejor decisión.

Finalmente, la metodología presentada en este capítulo puede ser aplicada en el análisis de cualesquiera otros sistemas (SPR o complejos) de los cuales dispongamos de algoritmos que evalúen la confiabilidad y funciones de costo. La dificultad de cada estudio en particular dependerá del grado de complejidad del sistema bajo análisis y del tipo de análisis que se desee efectuar (optimización topológica, análisis de sensibilidad, estudios comparativos, etc.)

A continuación presentamos las conclusiones y recomendaciones que se obtuvieron del presente trabajo.

CONCLUSIONES Y RECOMENDACIONES

En el presente trabajo se han analizado las principales características, funciones de costo y algoritmos de evaluación de confiabilidad más eficientes de SPR, así como también se efectúo una comparación de SPR en cuanto a confiabilidad y costo. En particular se describieron los sistemas K/N:G, CK/N:F, CCK/N:F y LCCS, los cuales constituyen los SPR más estudiados dentro de la literatura.

En los sistemas K/N:G, CK/N:F y CCK/N:F se manejan los conceptos de conectividad y capacidad. El concepto de conectividad se asocia a los llamados SPR no ponderados o sin restricciones de capacidad y el concepto de capacidad se asocia a los llamados SPR ponderados o con restricciones de capacidad.

En los SPR no ponderados, los tiempos de ejecución de los algoritmos de evaluación de confiabilidad WCH01, WCH02, KP01, WCH03, WCH04 y CCH01 dependen linealmente de los parámetros N y K del sistema. Si en un sistema SPR no ponderado aumentamos el número de componentes N en un factor de 10 (dejando invariable el parámetro K) el tiempo de ejecución de los algoritmos aumenta en un factor de 10. Por otro lado, los tiempos de ejecución de los algoritmos son semejantes ya que la estructura de los algoritmos es similar y requieren la misma cantidad de recursos de la computadora. Por todo lo anterior se concluye que un análisis comparativo de tiempos de ejecución para los algoritmos de SPR no ponderados no es relevante.

En los SPR ponderados se pudo determinar que los tiempos de ejecución de los algoritmos de evaluación de confiabilidad WCH02, WCH03 y CCH01 no dependen linealmente del parámetro W (capacidad de los componentes). En general, el efecto que produce el parámetro W sobre los tiempos de ejecución no ha sido estudiado aún con profundidad **[WC1]**, **[WC2]** y **[CCH]**. El análisis comparativo de tiempos de ejecución de los algoritmos para SPR ponderados se deja para cuando se propongan en la literatura nuevos algoritmos para estos sistemas.

Los algoritmos de los SPR ponderados sólo manejan números enteros para el parámetro W (capacidad de los componentes del sistema). Esta característica limita el espectro de aplicación de estos algoritmos en la práctica. Se necesita realizar una

modificación importante en los algoritmos para que estos abarquen a los SPR ponderados con parámetros de capacidad W no enteros.

En los sistemas LCCS se manejan los conceptos de componente biestado y multiestado. En los sistemas LCCS biestado los tiempos de ejecución de los algoritmos ZT01 y KP02 presentan las mismas características que los algoritmos de los sistemas K/N:G, CK/N:F y CCK/N:F por lo que un análisis comparativo de tiempos de ejecución no es relevante.

En cuanto a los sistemas LCCS multiestado, los autores consultados **[KP]** no estudian el efecto que se produce sobre el tiempo de ejecución del algoritmo KP02 al considerarse los componentes multiestado. El análisis comparativo de tiempos de ejecución para LCCS multiestado se deja para cuando se propongan en la literatura nuevos algoritmos para estos sistemas.

El título actual del trabajo refleja sólo uno de los aspectos del análisis realizado ya que además del estudio cuantitativo de los algoritmos se efectuó un análisis de confiabilidad y costo utilizando como elementos de comparación las funciones de confiabilidad y costo de los distintos SPR. Este análisis cualitativo nos muestra el efecto que producen la variación de los parámetros N, K y W en la confiabilidad y costo de los SPR y constituye una poderosa herramienta de diseño y análisis de SPR.

Los sistemas K/N:G, CK/N:F y CCK/N:F son una interesante alternativa de diseño con relación a los sistemas serie y paralelo, ya que estos sistemas pueden resultar más confiables que los sistemas serie y menos costosos que los sistemas paralelos. En particular, en los sistemas CK/N:F y CCK/N:F, un simple cambio en la posición de dos componentes de un sistema puede generar un sistema más confiable y menos costoso.

Con relación a los sistemas LCCS se analizó el efecto que se produce en la confiabilidad y el costo al variar la capacidad de los transmisores pero no se estudió el efecto de la reubicación de componentes debido principalmente a la gran cantidad de configuraciones posibles.

A partir del modelo general de **[SP]** se proponen los modelos particulares de costo que se emplearon en el análisis cualitativo de los sistemas K/N:G, CK/N:F, CCK/N:F y LCCS. Estos modelos no deben ser considerados como estándares sino solamente como herramientas para las fases iniciales de diseño de SPR ya que es muy difícil definir una función de costo exacta debido al hecho que no podemos anticipar todos los costos que puedan presentarse en una situación real.

Los algoritmos de evaluación de confiabilidad de SPR ponderados demostraron ser similares a los algoritmos de SPR no ponderados por lo tanto recomendamos los siguientes algoritmos para evaluar la confiabilidad de los sistemas K/N:G, CK/N:F y CCK/N:F ponderados y no ponderados:

WCH02 → Sistemas K/N:G ponderados y no ponderados

WCH03 → Sistemas CK/N:F ponderados y no ponderados

CCH01 → Sistemas CCK/N:F ponderados y no ponderados

Cuando se utilicen algoritmos de SPR ponderados en SPR no ponderados, la capacidad de los componentes del sistema debe ser la unidad $\Rightarrow W_i = 1$ con $1 \leq i \leq N$.

Los algoritmos de evaluación de confiabilidad de sistemas LCCS con componentes biestado demostraron ser similares a los algoritmos para sistemas LCCS con componentes multiestado por lo tanto recomendamos el siguiente algoritmo para evaluar la confiabilidad de sistemas LCCS:

KP02 → Sistemas LCSS componentes biestado y multiestado

Cuando se utilice el algoritmo de sistemas LCCS multiestado en sistemas LCCS biestado, la probabilidad de funcionamiento de los componentes deberá ser $p_{i,j} = 0$ para $0 \leq i \leq N$, $1 \leq j \leq k_i\text{-}1$.

Los sistemas complejos pueden contener subsistemas que deban representarse como SPR. Es totalmente válido determinar la confiabilidad del subsistema parcialmente redundante y sustituir éste por un elemento equivalente, el cual poseerá la confiabilidad

del subsistema original. A continuación se puede calcular la confiabilidad del nuevo sistema complejo por medio de un algoritmo, como por ejemplo el que se propone en **[M],** el cual evalúa la confiabilidad de sistemas complejos ponderados (con restricciones de capacidad) pero no maneja SPR.

Recomendamos integrar los algoritmos WCH02, WCH03, CCH01 y KP02 de SPR ponderados y sistemas LCCS multiestado a subrutinas de algoritmos que evalúen la confiabilidad de sistemas complejos ponderados o con restricciones de capacidad. La incorporación de un módulo que esté en la capacidad de manejar este tipo de SPR proporciona fortaleza y versatilidad a la evaluación de confiabilidad de sistemas complejos.

El análisis de confiabilidad y costo efectuado en este trabajo sobre los sistemas K/N:G, CK/N:F, CCK/N:F o LCCS debe ser complementando con análisis de disponibilidad. En **[NO]** se introduce el concepto de disponibilidad (probabilidad que el sistema esté funcionando en cualquier tiempo t) y MTBF (tiempo medio entre fallas) en el análisis de sistemas K/N:G.

Este análisis considera que (a) todos los componentes del sistema son idénticos (es decir que todos los componentes poseen la misma función de densidad de tiempo de falla), (b) definen la función de densidad como exponencial, (c) el sistema falla si y sólo si N-K+1 componentes fallan (siendo N: número de componentes y K: factor de redundancia parcial) y (d) existe reemplazo de los componentes que fallan.

REFERENCIAS BIBLIOGRÁFICAS

REFERENCIAS BIBLIOGRÁFICAS

[CCH] CHANG J., CHEN R., HWANG F., A Fast Reliability – Algorithm for the Circular Consecutive-Weighted-k-out-of-n:F System, IEEE Trans. Reliability, vol. 47, N° 4, 1998 December, pp 472-474.

[CFK] CHAO, M.; FU, J.; KOUTRAS, M, Survey of Reliabilities of consecutive-out-of-n:F & Related Systems, IEEE Transactions on Reliability, Vol. 44, N° 1, 1995 March, pp. 120-127.

[KP] KOSOW A., PREUSS W., Reliability of Linear Consecutively-Connected Systems with Multistate Components, IEEE Trans. Reliability, vol. 44, N° 3, 1995 September, pp 518-522.

[M] MARTINEZ L., Análisis de Confiabilidad y Disponibilidad de Sistemas con Restricciones de Capacidad en Enlaces y Nodos, Trabajo de Grado, Maestría de Investigación de Operaciones, UCV, Julio de 2002.

[MFJ] MILLER, I.; FREUND, J.; JOHNSON, R., Probabilidad y Estadística para Ingenieros, Prentice Hall, 4ta Edición, 1992

[NA] NACHLAS, J.; Fiabilidad. ISDEFE, 1ra Edición, 1995

[NO] NOWICKI, D.; Reliability Allocation with Partial Redundancy, 1991 Proceedings Annual Reliability and Maintainability Symposium, pp. 400-404

[P] PHAM, H.; On the Optimal Design of k-out-of-n:G Subsystems, IEEE Transactions on Reliability, Vol. 41, N° 4, 1992 December, pp. 572-574

[RSPK] RAI S., SARJE A. K., PRASAD E. V., KUMAR., Two Recursive Algorithms for Computing the Reliability of k-out-of-n Systems, IEEE Trans. Reliability, vol. R-36, N° 2, 1987 June, pp. 261-265.

[SFP] SFAKIANAKIS, M.; PAPASTAVRIDIS, S.; Reliability of a General Consecutive-k-out-of-n:F System, IEEE Transactions on Reliability, Vol. 42, N° 3, 1993 September, pp. 491-496

[SP] SUICH, R. C., PATTERSON, R. L., k-out-of-n Systems: Some Cost Considerations, IEEE Trans. Reliability, vol. 40, N° 3, 1991 August, pp. 259-264.

[WC1] WU J., CHEN R., An Algorithm for Computing the Reliability of Weighted-k-out-of-n Systems, IEEE Trans. Reliability, vol. 43, N° 2, 1994 June, pp 327-328.

[WC2] WU J., CHEN R., Efficient Algorithms for k-out-of-n & Consecutive-Weighted-k-out-of-n:F System, IEEE Trans. Reliability, vol. 43, N° 4, 1994 December, pp 650-655.

[WC3] WU J., CHEN R., An O(kn) Algorithm for a Circular Consecutive-k-out-of-n:F System, IEEE Trans. Reliability, vol. 41, N° 2, 1992 June, pp 303-305.

[YD] YOO Y. B., DEO N., A Comparison of Algorithms for Terminal-Pair Reliability, IEEE Trans. Reliability, vol. 37, N° 2, 1988 June, pp 210-215.

[ZUO] ZUO M., Reliability of Linear & Circular Consecutively-Connected Systems, IEEE Trans. Reliability, vol. 42, N° 3, 1993 September, pp 484-487.

APÉNDICE A

CÓDIGO DE LOS PROGRAMAS

Programa WCH01
Algoritmo WU-CHEN para sistemas K/N:G no ponderados

```
/*********************************************************/
/*Algoritmo WCH01 de Wu y Chen para sistemas K/N:G no
  ponderados
/*   VERSION 1.00 Programador: Manuel Rivera
/*********************************************************/

/* Este programa calcula la confiabilidad de un sistema K/N:G
   el cual es un sistema que esta funcionando si al menos K
   de sus N elementos están funcionando.

   El programa solicita los siguientes parámetros de entrada:
   K  : número mínimo de elementos funcionando requeridos
        para que el sistema esté funcionando
   n  : número de elementos que componen el sistema
   pi : probabilidad que el elemento i este funcionando

/*********************************************************/
#include <stdio.h>
#include <conio.h>
#include <math.h>
/*********************************************************/
#define Max 100
/*********************************************************/
void Lee_Datos(int &N,int &K,float p[Max])
{ int i;

     clrscr();
     printf(" PARÁMETROS DE ENTRADA DEL SISTEMA:\n =\n\n");
     do
     {     printf(" N :");
           scanf("%d", &N);    }
     while(N<=0 || N>=Max);
     do
     {    printf(" K :");
          scanf("%d",&K);      }
     while(K<=0 || K>N);
     printf(" PROBABILIDADES DE ÉXITO:\n  ==========\n\n");
     for (i=1; i<=N; i+=1)
     {    printf("   p(%d)=", i);
          scanf("%f",&p[i]);   }
}
/*********************************************************/
void Evaluar_R(int N,int K,float p[Max],float O[Max],float
I[Max],float B[Max])
```

```
{ int i, j;

     B[0] = 1;
     for (j=1;j<=K;j++)
     {    B[j] = 0;      }
     for (i=1;i<=N;i++)
     {    I[0] = 1;
          for (j=1; j<=K; j++)
          {       O[j] = p[i] * I[j-1] + (1-p[i]) * B[j];
                  I[j] = B[j];
                  B[j] = O[j];
          }
     }
}
/*********************************************************/
void Escribir_Resultados(int N,int K,float B[Max])
{
     clrscr();
     printf("La confiabilidad del sistema %d de %d:G es
     \n",K,N);
     printf("R = %f\n",B[K]);
}
/*********************************************************/
void Detenhastaque1(void)
{    int c;

     printf("\n\nIntroduzca 1 y presione ENTER para salir:");
     do
     c=getchar();
     while(c!=49);
}
/*********************************************************/
void main()
{
     int N;
     int K;
     float p[Max];
     float O[Max];
     float I[Max];
     float B[Max];

     Lee_Datos(N,K,p);
     Evaluar_R(N,K,p,O,I,B);
     Escribir_Resultados(N,K,B);
     Detenhastaque1();
}
/*********************************************************/
```

Programa WCH02

Algoritmo WU-CHEN para sistemas K/N:G ponderados

```
/*********************************************************/
/*   Algoritmo WCH02 de Wu y Chen para sistemas K/N:G
     ponderados
/*   VERSION 1.00 Programador: Manuel Rivera
/*********************************************************/

/* Este programa calcula la confiabilidad de un sistema K de
   N:G ponderado el cual es un sistema que esta funcionando
   si el peso total de los elementos que están funcionando es
   de al menos K

   El programa solicita los siguientes par metros de entrada:
   K : peso total mínimo de todos los elementos funcionando
   el cual hacen que el sistema esté funcionando
   N : número de elementos que componen el sistema
   wi : peso del componente i
   pi: probabilidad que el elemento i este funcionando

/*********************************************************/
#include <stdio.h>
#include <conio.h>
#include <math.h>
/*********************************************************/
#define Max 100
/*********************************************************/
void Lee_Datos(int &N,int &K,float p[Max],float w[Max])
{ int i;
     clrscr();
     printf("  PARÁMETROS DE ENTRADA:\n ============\n\n");
     do
     {     printf(" N :");
           scanf("%d",&N);     }
     while(N<=0 || N>=Max);
     do
     {    printf(" K :");
          scanf("%d",&K);      }
     while(K<=0);
     printf("  PESOS wi:\n  ==================\n\n");
     for (i=1; i<=N; i+=1)
               {     printf("   w(%d)=", i);
                     scanf("%f",&w[i]);      }

     printf("  PROBABILIDADES DE ÉXITO:\n ===========\n\n");
```

```
     for (i=1; i<=N; i+=1)
               {         printf("   p(%d)=", i);
                         scanf("%f",&p[i]);            }
}
/*********************************************************/
void Evaluar_Rij(int N,int K,float p[Max],float w[Max],float
R[Max][Max])
{ int i, j;

     for (i=0; i<=N; i+=1) { R[i][0] = 1; }
     for (j=1; j<=K; j+=1) { R[0][j] = 0; }
     for (i=1; i<=N; i+=1)
     {     for (j=1; j<=K; j+=1)
           {  if(j - w[i]>=0) R[i][j] = p[i]*R[i-1][j-w[i]] +
                                        (1-p[i])*R[i-1][j];
              else R[i][j] = p[i] + (1-p[i])*R[i-1][j];
           }
     }
}
/*********************************************************/
void Escribir_Resultados(int N,int K,float R[Max][Max])
{
     clrscr();
     printf("La confiabilidad del sistema %d de %d:G
     ponderado es %f\n",K,N,R[N][K]);
}
/*********************************************************/
void Detenhastaque1(void)
{    int c;

     printf("\n\nIntroduzca 1 y presione ENTER para salir:");
     do
     c=getchar();
     while(c!=49);

}
/*********************************************************/
void main()
{
     int N;
     int K;
     float p[Max];
     float w[Max];
     float R[Max][Max];

     Lee_Datos(N,K,p,w);
     Evaluar_Rij(N,K,p,w,R);
```

```
     Escribir_Resultados(N,K,R);
     Detenhastaque1();
}
/*************************************************************/
```

Programa KP01

Algoritmo KOSOW-PREUSS para sistemas CK/N:F no ponderados

```
/*************************************************************/
/*   Algoritmo KP01 de Kosow y Preuss para sistemas CK/N:F no
     ponderados
/*   VERSION 1.00 Programador: Manuel Rivera
/*************************************************************/

/* Este programa calcula la confiabilidad de un sistema K de
N:F consecutivo lineal el cual es un sistema que consiste de
una secuencia de N componentes ordenados en una línea tal que
el sistema no está funcionando si al menos K componentes
consecutivos fallan.

El programa solicita los siguientes par metros de entrada:
K : peso total mínimo de los componentes consecutivos
fallados los cuales causan que el sistema falle
N : número de componentes del sistema
pi: probabilidad que el elemento i este funcionando

/*************************************************************/
#include <stdio.h>
#include <conio.h>
#include <math.h>
/*************************************************************/
#define Max 100
/*************************************************************/
void Lee_Datos(int &N,int &K,float p[Max],float q[Max])
{ int i, j;
     clrscr();
     printf(" PARÁMETROS DE ENTRADA DEL SISTEMA:\n ===\n\n");
     printf(" NÚMERO DE COMPONENTES (NO INCLUIR FUENTE Y
     CARGA):\n  ================\n\n");
     do
     {     printf(" N :");
           scanf("%d", &N);     }
     while(N<0 || N>=Max);


     printf(" \nCAPACIDAD DE TRANSMISIÓN K :\n ==\n\n");
```

```
     do
     {     printf("  K =");
           scanf("%d",&K);       }
     while(K<=0);
     printf(" PROBABILIDADES DE ÉXITO:\n  ============\n\n");
     for (i=1; i<=N; i+=1)
     {   printf("   p(%d)=",i);
         scanf("%f",&p[i]);      }
     for (i=1; i<=N; i+=1)
               q[i] = 1 - p[i];
}
/*********************************************************/
float fq(int N,int K, float q[Max])
{ int r;
     float T;

     T = 1;
     for(r=0;r<=K-1;r++)
     T *= q[N-r];
     return T;
}
/*********************************************************/
void Evaluar_R(int N,int K,float p[Max],float q[Max],float
R[Max])
{ int j,r;
     float H, T;

     for(j=0;j<K;j++) R[j] = 1;
     T = 1;
     for(r=1;r<=K;r++)
     T *= q[r];
     R[K] = 1 - T;
     for(j=K+1;j<=N;j++)
     R[j] = R[j-1] - p[j - K] * fq(N,K,q) * R[j - K - 1];
}
/*********************************************************/
void Escribir_Resultados(int N,int K,float R[Max])
{
     clrscr();
     printf("La confiabilidad del sistema C%d/%d es:
     \n",K,N);
     printf("%f\n",R[N]);
}
/*********************************************************/
void Detenhastaque1()
{    int c;
```

```
     printf("\n\nIntroduzca 1 y presione ENTER para salir:");
     do
     c=getchar();
     while(c!=49);

}
/*************************************************************/
void main()
{
     float p[Max], q[Max];
     int N, K;
     float R[Max];

     Lee_Datos(N,K,p,q);
     Evaluar_R(N,K,p,q,R);
     Escribir_Resultados(N,K,R);
     Detenhastaque1();
}
/*************************************************************/
```

Programa WCH03

Algoritmo WU-CHEN para sistemas CK/N:F ponderados

```
/*************************************************************/
/*   Algoritmo WCH03 de Wu y Chen para sistemas CK/N:F
     ponderados
/*   VERSION 1.00 Programador: Manuel Rivera
/*************************************************************/

/* Este programa calcula la confiabilidad de un sistema k de
N:F ponderado consecutivo lineal el cual es un sistema que
consiste de una secuencia de N componentes ordenados en una
línea tal que el sistema no está funcionando si al menos K
componentes consecutivos fallan.

El programa solicita los siguientes par metros de entrada:
K : peso total mínimo de los componentes consecutivos
fallados los cuales causan que el sistema falle
N : número de componentes del sistema
wi: peso del componente i
qi: probabilidad que el elemento i este fallado

/*************************************************************/
#include <stdio.h>
#include <conio.h>
#include <math.h>
```

```
/*********************************************************/
#define Max 100
/*********************************************************/
void Lee_Datos(int &N,int &K,float p[Max],float q[Max],float
w[Max])
{
     int i;

     clrscr();
     printf("  PARÁMETROS DE ENTRADA DEL SISTEMA:\n  =\n\n");
     do
     {     printf(" N :");
           scanf("%d", &N);      }
     while(N<=0 || N>=Max);
     do
     {    printf(" K :");
          scanf("%d",&K);        }
     while(K<=0);
     printf("  PESOS wi:\n  ==================\n\n");
     for (i=1; i<=N; i+=1)
     {     printf("   w(%d)=", i);
           scanf("%f",&w[i]);    }
     printf("  PROBABILIDADES DE FALLA:\n ==========\n\n");
     for (i=1; i<=N; i+=1)
     {     printf("   q(%d)=", i);
           scanf("%f",&q[i]);    }
      for (i=1; i<=N; i+=1)
                  p[i] = 1 - q[i];
}
/*********************************************************/
void Algoritmo_A(int N,int K,int &m,float q[Max],float
w[Max],float Wet[Max],float Q[Max],int Beg[Max],int End[Max])
{ int event, component;

     m = 0; event = 1;
     Wet[event] = 0; Q[event] = 1; Beg[event] = 1;

     for (component=1;component<=N;component+=1)
     {
           Wet[event] += w[component];
           Q[event] *= q[component];
           if (Wet[event] >= K)
           {
                    m += 1;
                    End[event] = component;
                    while(Wet[event] - w[Beg[event]] >= K)
                    {
```

```
                              Wet[event] -= w[Beg[event]];
                              Q[event] /= q[Beg[event]];
                              Beg[event] += 1;
                        }
                        Beg[event + 1] = Beg[event] + 1;
                        Wet[event + 1] = Wet[event]-
                                         w[Beg[event]];
                        Q[event + 1] = Q[event]/q[Beg[event]];
                        event += 1;
                }
        }
}
/*************************************************************/
float H1(int j,int i,float q[Max],int Beg[Max])
{ int l;
     float H;

     H=1;
     for (l=Beg[j-1]+i+1;l<=Beg[j]-1;l+=1)
     {  H *= q[l];  }
     return H;
}
/*************************************************************/
void Evaluar_Fl(int m,int N,float p[Max],float q[Max],int
Beg[Max],int End[Max],float Q[Max],float FL[Max])
{ int j, i;
     float H;

     for (j=0;j<=End[1]-1;j+=1)
     { FL[j] = 0;  }
     FL[End[1]] = Q[1];
     for (j=2;j<=m;j+=1)
           {         H = 0;
                     for (i=0;i<=(Beg[j]-Beg[j-1]-1);i+=1)
                     {  H+=(1-FL[Beg[j-1]+i-1])*p[Beg[j-
                            1]+i]*H1(j,i,q,Beg)*Q[j];  }
                     FL[End[j]] = H + FL[End[j-1]];
           }
     for(j=End[m]+1;j<=N;j+=1)
     { FL[j] = FL[End[m]];    }
}
/*************************************************************/
void Escribir_Resultados(int N,int K,float FL[Max],int
Beg[Max],int End[Max])
{ int j;
     clrscr();
```

```
     printf("La confiabilidad del sistema %d de %d:F
     ponderado consecutivo es \n",K,N);
     printf("R = %f\n",1-FL[N]);
}
/*********************************************************/
void Detenhastaque1(void)
{    int c;

     printf("\n\nIntroduzca 1 y presione ENTER para salir:");
     do
     c=getchar();
     while(c!=49);
}
/*********************************************************/
void main()
{
     int N, K, m;
     float p[Max];    //Vector de probabilidades de éxito
     float q[Max];    //Vector de probabilidades de falla
     float w[Max];    //Vector de pesos
     int Beg[Max];    //Vector de primeros componentes de Si
     int End[Max];    //Vector de últimos componentes de Si
     float Wet[Max];  //Vector de pesos totales de Si
     float Q[Max];    //Vector de productos de qi
     float FL[Max];   //Vector de "no confiabilidad"


     Lee_Datos(N,K,p,q,w);      //Lee los datos del problema
     Algoritmo_A(N,K,m,q,w,Wet,Q,Beg,End); //Encuentra los
                                            eventos Si
     Evaluar_Fl(m,N,p,q,Beg,End,Q,FL);    //Evalúa la "no
                                          confiabilidad" del
                                          problema
     Escribir_Resultados(N,K,FL,Beg,End);  //Escribe los
                                          resultados
                                          obtenidos
     Detenhastaque1();
}
/*********************************************************/
```

Programa WCH04

Algoritmo WU-CHEN para sistemas CCK/N:F no ponderados

```
/*********************************************************/
/*   Algoritmo WCH04 de Wu y Chen para sistemas CCK/N:F no
     ponderados
/*   VERSION 1.00 Programador: Manuel Rivera
/*********************************************************/

/* Este programa calcula la confiabilidad de un sistema k de
N:F consecutivo circular el cual es un sistema que consiste
de una secuencia de N componentes ordenados en un circulo tal
que el sistema no está  funcionando si al menos K componentes
consecutivos fallan.

El programa solicita los siguientes par metros de entrada:
N : número de componentes del sistema
K : número mínimo de componentes consecutivos fallados los
cuales causan que el sistema falle
qi: probabilidad que el elemento i este fallado

/*********************************************************/
#include <stdio.h>
#include <conio.h>
#include <math.h>
/*********************************************************/
#define Max 100
/*********************************************************/
void Lee_Datos(int &N,int &K,float q[Max])
{
     int i;

     clrscr();
     printf("  PARÁMETROS DE ENTRADA DEL SISTEMA:\n  =\n\n");
     do
     {     printf(" N :");
           scanf("%d", &N);        }
     while(N<=0 || N>=Max);
     do
     {    printf(" K :");
          scanf("%d",&K);          }
     while(K<=0);
     printf("  PROBABILIDADES DE FALLA:\n ==========\n\n");
     for (i=1; i<=N; i+=1)
     {    printf("   q(%d)=", i);
          scanf("%f",&q[i]);       }
```

```
}
/*************************************************************/
float fq(int i,int component,int N,float q[Max])
{   if(component<=i || component>N)
          {    return 1;          }
          else
          {      return q[component];       }
}
/*************************************************************/
float H1(int j,int i,float q[Max],int Beg[Max])
{ int l;
     float H;

     H=1;
     for (l=Beg[j-1]+i+1;l<=Beg[j]-1;l+=1)
     {  H *= q[l];  }
     return H;
}
/*************************************************************/
float Fsys(int i,int n,int N,int K,int m,float q[Max],int
Beg[Max],int End[Max],float Wet[Max],float Q[Max],float
FL[Max])
{ int event, component;

   m = 0; event = 1;
   Wet[event] = 0; Q[event] = 1; Beg[event] = 1;

   for (component=1;component<=n;component+=1)
   { Wet[event] += 1;
     Q[event] *= fq(i,component,N,q);
     if (Wet[event] >= K)
     {  m += 1;
        End[event] = component;
        while(Wet[event] - 1 >= K)
        {   Wet[event] -= 1;
            Q[event] /= fq(i,Beg[event],N,q);
            Beg[event] += 1;             }
            Beg[event + 1] = Beg[event] + 1;
            Wet[event + 1] = Wet[event] - 1;
            Q[event + 1] = Q[event]/fq(i,Beg[event],N,q);
            event += 1;       }
      }

   int j, t;
   float H;

   for (j=0;j<=End[1]-1;j+=1)
```

```
   { FL[j] = 0;  }
     FL[End[1]] = Q[1];
     for (j=2;j<=m;j+=1)
     {    H = 0;
          for (t=0;t<=(Beg[j] - Beg[j-1] - 1);t+=1)
          {  H+=(1-FL[Beg[j-1]+t-1])*(1-fq(i,Beg[j-
                1]+t,N,q))*H1(j,t,q,Beg)*Q[j];   }
           FL[End[j]] = H + FL[End[j-1]];
     }
     for(j=End[m]+1;j<=n;j+=1)
     { FL[j] = FL[End[m]];    }

     return FL[n];
}
/*********************************************************/
void Evaluar_R(int N,int K,int &m,float q[Max],int
Beg[Max],int End[Max],float Wet[Max],float Q[Max],float
FL[Max],float &R)
{ int i, j;
  float H, T;

  H = Fsys(0,N,N,K,m,q,Beg,End,Wet,Q,FL);

  for(i=1;i<=K-1;i++)
  {   T = Fsys(i,N+i,N,K,m,q,Beg,End,Wet,Q,FL) - Fsys(i,N+i-
          1,N,K,m,q,Beg,End,Wet,Q,FL);
          for(j=1;j<=i;j++)
          {      T *= q[j];        }
          H += T;
  }
  R = 1 - H;
}
/*********************************************************/
void Escribir_Resultados(int N,int K,float R)
{ int j;
     clrscr();
     printf("La confiabilidad del sistema %d de %d:F
     consecutivo circular es \n",K,N);
     printf("R = %f\n",R);
}
/*********************************************************/
void Detenhastaque1(void)
{    int c;

     printf("\n\nIntroduzca 1 y presione ENTER para salir: );
     do
     c=getchar();
```

```
     while(c!=49);
}
/*************************************************************/
void main()
{
     int N, K, m;
     float R;
     float q[Max];    //Vector de probabilidades de falla
     int Beg[Max];    //Vector de primeros componentes de Si
     int End[Max];    //Vector de ultimos componentes de Si
     float Wet[Max];  //Vector de pesos totales de Si
     float Q[Max];    //Vector de productorias de qi
     float FL[Max];   //Vector de "no confiabilidad"

     Lee_Datos(N,K,q);//Lee los datos del problema
     Evaluar_R(N,K,m,q,Beg,End,Wet,Q,FL,R);  //Evaluar la
                                              confiabilidad
                                              del sistema
     Escribir_Resultados(N,K,R);    //Escribe los resultados
                                    obtenidos
     Detenhastaque1();
}
/*************************************************************/
```

Programa CCH01

Algoritmo CHAN-CHEN-HWANG para sistemas CCK/N:F ponderados

```
/*************************************************************/
/*   Algoritmo CCH01 de Chang, Chen y Hwang para sistemas
CCK/N:F ponderados
/*   VERSION 1.00 Programador: Manuel Rivera
/*************************************************************/

/* Este programa calcula la confiabilidad de un  sistema K de
N:F consecutivo ponderado circular el cual es un sistema que
consiste de una secuencia  de N componentes ordenados en un
circulo tal que el sistema no está  funcionando si el peso
total de algunos componentes consecutivos es de al menos K.
Los componentes deben estar previamente ordenados por el
usuario de tal forma que el componente 1 posee la mayor
capacidad.

El programa solicita los siguientes par metros de entrada:
K : peso total mínimo de los componentes consecutivos
fallados los cuales causan que el sistema falle
N : número de componentes del sistema
```

```
wi : peso del componente i
qi: probabilidad que el elemento i este fallado

/*********************************************************/
#include <stdio.h>
#include <conio.h>
#include <math.h>
/*********************************************************/
#define Max 100
/*********************************************************/
void Lee_Datos(int &N,int &K,float p[Max],float q[Max],float
w[Max])
{
     int i;

     clrscr();
     printf("  PARÁMETROS DE ENTRADA DEL SISTEMA:\n =\n\n");
     do
     {     printf(" N :");
           scanf("%d", &N);     }
     while(N<=0 || N>=Max);
     do
     {    printf(" K :");
          scanf("%d",&K);       }
     while(K<=0);
     printf("  PESOS wi:\n  ==================\n\n");
     for (i=1; i<=N; i+=1)
     {     printf("   w(%d)=", i);
           scanf("%f",&w[i]);   }
     printf("  PROBABILIDADES DE FALLA:\n  ==========\n\n");
     for (i=1; i<=N; i+=1)
     {    printf("   q(%d)=", i);
          scanf("%f",&q[i]);     }
      for (i=1; i<=N; i+=1)
          p[i] = 1 - q[i];
}
/*********************************************************/
void Calcular_T(int N,int K,int &T,float w[Max])
{ int i, j;
     float H;

     T = 1;
     for (i=1;i<=N+1;i+=1)
     {   H = 0;
         for (j=1;j<=i-1;j+=1)
         { H += w[j]; }
           if (H<K && T < i) T = i;       }
```

```
}
/*************************************************************/
float qj(int i,float q[Max])
{ int j;
     float H;

     H = 1;
     for (j = 1; j<= i-1; j+=1)
     {  H *= q[j];   }
     return H;
}

/*************************************************************/
float fp(int i,int N,float p[Max])
{ if(i<=N) {    return p[i];   }
     else  {    return 0;      }
}
/*************************************************************/
float fq(int i,int N,float q[Max])
{ if(i<=N) {   return q[i];   }
     else  {   return 1;    }
}
/*************************************************************/
float fw(int i,int N,float w[Max])
{ if(i<=N) {  return w[i];  }
     else  {  return 1;     }
}
/*************************************************************/
float H1(int j,int i,float q[Max],int Beg[Max],int N)
{ int l;
     float H;

     H=1;
     for (l=Beg[j-1]+i+1;l<=Beg[j]-1;l+=1)
     {  H *= fq(l,N,q);  }
     return H;
}
/*************************************************************/
float Fl(int i,int j,int N,int K,int &m,float p[Max],float
q[Max],float w[Max],int Beg[Max],int End[Max],float
Wet[Max],float Q[Max],float FL[Max])
{ int event, component;

   if(j-i+1 < K) {  return 0;  }
   else  {    m = 0; event = 1;
              Wet[event] = 0; Q[event] = 1; Beg[event] = i;
```

```
                for (component=i;component<=j;component+=1)
                {
                   Wet[event] += fw(component,N,w);
                   Q[event] *= fq(component,N,q);
                   if (Wet[event] >= K)
                   {
                      m += 1;
                      End[event] = component;
                      while(Wet[event]-fw(Beg[event],N,w) >= K)
                      {
                         Wet[event] -= fw(Beg[event],N,w);
                         Q[event] /= fq(Beg[event],N,q);
                         Beg[event] += 1;
                      }
                      Beg[event + 1] = Beg[event] + 1;
                      Wet[event + 1] = Wet[event] -
                                       fw(Beg[event],N,w);
                      Q[event + 1]=Q[event]/fq(Beg[event],N,q);
                      event += 1;         }
                   }
                 }

     int g, h;
     float H;

     for (g=0;g<=End[1]-1;g+=1)
     { FL[g] = 0;  }
     FL[End[1]] = Q[1];

     for (g=2;g<=m;g+=1)
     {  H = 0;
        for (h=0;h<=(Beg[g] - Beg[g-1] - 1);h+=1)
        {  H+=(1-FL[Beg[g-1]+h-1])*fp(Beg[g -
              1]+h,N,p)*H1(g,h,q,Beg,N)*Q[g];     }
           FL[End[g]] = H + FL[End[g-1]];
     }

     for(g=End[m]+1;g<=j;g+=1)
     { FL[g] = FL[End[m]];    }

     return FL[j];
}
/*********************************************************/
float Evaluar_Rc(int N,int K,int T,int &m,float p[Max],float
q[Max],float w[Max],int Beg[Max],int End[Max],float
Wet[Max],float Q[Max],float FL[Max])
{ int i;
```

```
  float H;

  H = 0;
  for (i=1;i<=T;i+=1)
  {  H += (1 - Fl(i+1,N+i-1,N,K,m,p,q,w,Beg,End,Wet,Q,FL)) *
         qj(i,q) * p[i];
  }
  return H;
}
/*********************************************************/
void Escribir_Resultados(int N,int K,float R)
{ int j;
     clrscr();
     printf("La confiabilidad del sistema %d de %d:F
     ponderado consecutivo circular es \n",K,N);
     printf("R = %f\n",R);
}
/*********************************************************/
void Detenhastaque1(void)
{    int c;

     printf("\n\nIntroduzca 1 y presione ENTER para salir:");
     do
     c=getchar();
     while(c!=49);
}
/*********************************************************/
void main()
{
     int N, K, m, T;
     float R;
     float w[Max];   //Vector de pesos
     float q[Max];   //Vector de probabilidades de falla
     float p[Max];   //Vector de probabilidades de exito
     int Beg[Max];   //Vector de primeros componentes de Si
     int End[Max];   //Vector de ultimos componentes de Si
     float Wet[Max]; //Vector de pesos totales de Si
     float Q[Max];   //Vector de productorias de qi
     float FL[Max];  //Vector de "no confiabilidad"

     Lee_Datos(N,K,p,q,w);  //Lee los datos del problema
     Calcular_T(N,K,T,w);   //Calcula T

     if (T>N) {   R = 1;   }
     else {   R = Evaluar_Rc(N,K,T,m,p,q,w,Beg,End,Wet,Q,FL);
          }  //Calcula Rc(1,N)
```

```
     Escribir_Resultados(N,K,R);     //Escribe los resultados
                                      obtenidos
     Detenhastaque1();
}
/*********************************************************/
```

Programa ZT01

Algoritmo ZUO para sistemas LCCS biestado

```
/*********************************************************/
/*   Algoritmo ZT01 de Zuo para sistemas LCCS biestado:
/*   VERSION 1.00 Programador: Manuel Rivera
/*********************************************************/

/* Este programa calcula la confiabilidad de un  sistema
LCCS. Un sistema consistente de una fuente(0),N componentes
(1,2,...N) y una carga(N+1) es un sistema CCS si la fuente
esta conectada a los componentes {1,2,...,k0} y los
componentes j {1<=j<=N} están conectados a los componentes
{j+1,j+2,...,j+kj} por arcos. El sistema funciona si y solo
si hay una conexión de la fuente a la carga a través de
componentes que funcionan.
El programa solicita los siguientes parámetros de entrada:
N : número de componentes del sistema (incluyendo fuente y
carga)
ki : capacidad de transmisión del componente i
pi: probabilidad que el elemento i este funcionando

/*********************************************************/
#include <stdio.h>
#include <conio.h>
#include <math.h>
/*********************************************************/
#define Max 100
/*********************************************************/
void Lee_Datos(int &N,int k[Max],float p[Max])
{ int i;

     clrscr();
     printf(" PARÁMETROS DE ENTRADA DEL SISTEMA:\n  ==\n\n");
     printf(" NÚMERO DE COMPONENTES, INCLUYENDO FUENTE Y
     CARGA:\n  ==================\n\n");
     do
     {     printf(" N :");
           scanf("%d", &N);    }
```

```
     while(N<=0 || N>=Max);
     printf(" CAPACIDAD DE TRANSMISIÓN Ki:\n ===\n\n");
     for (i=0; i<N-1; i+=1)
     {  printf("   k(%d)=", i);
        scanf("%d",&k[i]);      }
     printf("  PROBABILIDADES DE ÉXITO:\n  ==========\n\n");
     for (i=1; i<N-1; i+=1)
     {    printf("   p(%d)=", i);
          scanf("%f",&p[i]);   }
}
/*********************************************************/
void Evaluar_R(int N,int k[Max],float p[Max],float R[Max])
{ int i,j;
     float temp;

     for (i=N-2;i>=0;i-=1)
     {  if(i+k[i]>=N-1)
           R[i] = 1;
           else {   R[i] = 0;
                    temp = 1;
                    for (j=i+1;j<=i+k[i];j+=1)
                    {  if(j+k[j]>i+k[i])
                       { R[i] += temp*p[j]*R[j];
                                 temp *=(1 - p[j]);       }
                    }
                }
     }
}
/*********************************************************/
void Escribir_Resultados(int N,float R[Max])
{
     clrscr();
     printf("La confiabilidad del sistema LCCS de %d
     componentes es: \n",N);
     printf("R = %f\n",R[0]);
}
/*********************************************************/
void Detenhastaque1(void)
{    int c;

     printf("\n\nIntroduzca 1 y presione ENTER para salir:");
     do
     c=getchar();
     while(c!=49);

}
```

```
/*********************************************************/
void main()
{
     int N;
     int k[Max];
     float p[Max];
     float R[Max];

     Lee_Datos(N,k,p);
     Evaluar_R(N,k,p,R);
     Escribir_Resultados(N,R);
     Detenhastaque1();
}
/*********************************************************/
```

Programa KP02

Algoritmo KOSOW-PREUSS para sistemas LCCS multiestado

```
/*********************************************************/
/*   Algoritmo KP02 de Kosow y Preuss para sistemas LCCS
     multiestado
/*   VERSION 1.00 Programador: Manuel Rivera
/*********************************************************/

/* Este programa calcula la confiabilidad de un sistema
LCCSMC. Un sistema lineal consecutivamente conectado con
componentes multiestado LCCSMC (por sus siglas en inglés)
consiste de n+2 componentes Ci multiestado estadísticamente
independientes ordenados en una línea; el nodo fuente es Co,
el nodo terminal es Cn+1, éste último se considera totalmente
confiable.
El sistema falla si y solo si no hay conexión desde Co hasta
Cn+1.

El programa solicita los siguientes parámetros de entrada:
N : número de componentes del sistema (sin incluir nodo
fuente y nodo terminal)
ki: capacidad de transmisión máxima del componente i
pi,j: probabilidad que el componente i esta en el estado j

/*********************************************************/
#include <stdio.h>
#include <conio.h>
#include <math.h>
/*********************************************************/
#define Max 100
```

```
/*********************************************************/
void Lee_Datos(int &N,int k[Max],float p[Max][Max])
{ int i, j;
     clrscr();
     printf(" PARÁMETROS DE ENTRADA DEL SISTEMA:\n  =\n\n");
     printf(" NÚMERO DE COMPONENTES, NO INCLUIR FUENTE Y
     CARGA:\n  ==================\n\n");
     do
     {     printf(" N :");
           scanf("%d", &N);        }
     while(N<0 || N>=Max);
     printf(" \nCAPACIDAD DE TRANSMISIÓN MÁXIMA Ki:\n
     ==\n\n");
     for (i=0; i<=N; i+=1)
     {
          printf("   k(%d)=", i);
          scanf("%d",&k[i]);       }
     printf(" PROBABILIDADES Pij:\n  ==============\n\n");
     for (i=0; i<=N; i+=1)
     for (j=0;j<=k[i];j++)
     {   printf("   p(%d)(%d)=",i,j);
         scanf("%f",&p[i][j]);     }
}
/*********************************************************/
int maximo(int N,int k[Max])
{  int i, P;

      P=0;
      for(i=0;i<=N;i++)
      {     if(k[P]<k[i]) P = i;         }
      return k[P];
}
/*********************************************************/
int minimo(int a,int b)
{
      if(a<=b) return a;
      else     return b;
}
/*********************************************************/
float A(int v, int N,float p[Max][Max],int k[Max])
{   int j, r, s;
          float H, P;

          if(v==1) {  return 1 - p[N][0];     }
          if(v==2) {   H = 0;
                       for(j=2;j<=k[N-1];j++)
                       {       H += p[N-1][j];          }
```

```
                        return p[N][0] * H;
                    }
          if(v>=3) {    P = 1;
                        for(r=1;r<=v-2;r++)
                        {  H = 0;
                           for(s=0;s<=minimo(r,k[N-r]);s++)
                           {    H += p[N-r][s];        }
                           P *= H;
                        }
                   H = 0;
                   for(j=v;j<=k[N-v+1];j++)
                   {     H += p[N-v+1][j];             }
                   return p[N][0] * P * H;
                   }
          return 0;
}
/*************************************************************/
float Rk(int N,int K,int k[Max],float p[Max][Max],float
R[Max])
{   int j, v;

     K = maximo(N,k);
     R[0] = 1 - p[0][0];
     j = 1;
     for(j=1;j<k[0];j++)
     {      R[j] = 0;
            for(v=1;v<=j;v++)
            {      R[j] += A(v,j,p,k) * R[j-v];         }
            R[j] += A(j+1,j,p,k);
     }
     for(j=k[0];j<=N;j++)
     {      R[j] = 0;
            for(v=1;v<=minimo(K,N);v++)
            {      R[j] += A(v,j,p,k) * R[j-v];         }

     }
      return R[N];
}
/*************************************************************/
void Evaluar_R(int N,int &K,int k[Max],float
p[Max][Max],float R[Max])
{ int v;
     float H, T;

     if(N==0) R[0] = 1 - p[0][0];
     else     R[N] = Rk(N,K,k,p,R);
}
```

```
/*********************************************************/
void Escribir_Resultados(int N,float R[Max])
{
     clrscr();
     printf("La confiabilidad del sistema LCCSMC de %d
     componentes es: \n",N);
     printf("%f\n",R[N]);
}
/*********************************************************/
void Detenhastaque1()
{    int c;

     printf("\n\nIntroduzca 1 y presione ENTER para salir:");
     do
     c=getchar();
     while(c!=49);
}
/*********************************************************/
void main()
{
     int k[Max];
     float p[Max][Max];
     int N, K;
     float R[Max];

     Lee_Datos(N,k,p);
     Evaluar_R(N,K,k,p,R);
     Escribir_Resultados(N,R);
     Detenhastaque1();
}
/*********************************************************/
```

APÉNDICE B

EJEMPLOS DE SISTEMAS PARCIALMENTE REDUNDANTES

Sistemas K/N:G no ponderados:

Considere la fig. 1.2:

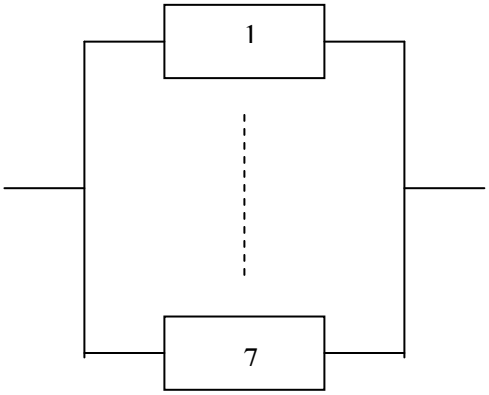


Fig. 1.2 Sistema 4/7:G no ponderado

Esta figura corresponde al DBC del sistema de calentamiento de un reactor químico, formado por 7 resistencias eléctricas. El diseñador del proceso nos dice que si al menos funcionan 4 de las 7 resistencias del sistema de calentamiento, la temperatura a la que ocurre la reacción permanecerá dentro de valores permitidos y el reactor funciona de manera adecuada. Supongamos los siguientes eventos:

Evento 1: Funcionan 6 resistencias. El sistema funciona ya que la temperatura se mantiene por encima de cierto valor permitido y el proceso no se detiene.

Evento 2: Funcionan 3 resistencias. El sistema no funciona ya que la temperatura desciende por debajo de cierto valor permitido y el proceso se detiene por baja temperatura.

Evento 3: Funcionan 4 resistencias. El sistema funciona ya que la temperatura se mantiene justo por encima de cierto valor permitido y el proceso no se detiene.

Sistemas K/N:G ponderados:

Considere la fig. 1.3

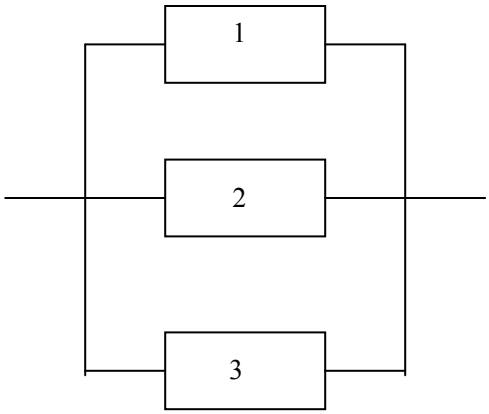


Fig. 1.3 Sistema 30/3:G ponderado

La fig. 1.3 representa el DBC de un sistema de 3 servidores encargados de manejar los requerimientos de memoria del correo electrónico de cierto número de usuarios. El servidor 1 maneja 10 GB, el servidor 2 maneja 20 GB y el servidor 3 maneja 20 GB.

La persona encargada del servicio nos dice que se necesita disponer en todo momento de al menos 30 GB de memoria para que todos los usuarios revisen adecuadamente su correo. Supongamos los siguientes eventos:

Evento 1: Funciona servidor 1 (10 GB), funciona servidor 2 (20 GB) y falla el servidor 3 (20 GB). El sistema funciona ya que tenemos 30 GB de memoria disponible.

Evento 2: Funciona servidor 1 (10 GB), falla servidor 2 (20 GB) y falla el servidor 3 (20 GB). El sistema no funciona ya que tenemos sólo 10 GB de memoria disponible.

Evento 3: Falla servidor 1 (10 GB), falla servidor 2 (20 GB) y funciona el servidor 3 (20 GB). El sistema no funciona ya que tenemos sólo 20 GB de memoria disponible.

Sistemas CK/N:F no ponderados:

Considere la fig. 1.4

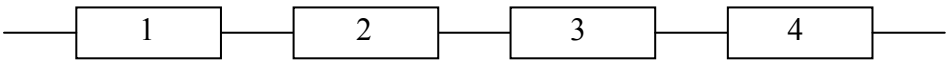


Fig. 1.4 Sistema C2/4:F no ponderado

La fig. 1.4 representa el DBC del sistema de termo-retracción de una máquina empacadora. El sistema está constituido por 4 hornos eléctricos conectados en serie, a través de ellos pasa una banda transportadora que lleva grupos de botellas envueltas en un papel que se encoge y endurece con la temperatura.

El diseñador de la máquina nos dice que si tenemos dos o más hornos consecutivos fallados el papel que envuelve al grupo de botellas no comprime bien y el paquete tendrá que envolverse de nuevo, es decir el sistema no funciona. Supongamos los siguientes eventos:

Evento 1: Funciona el horno 1, funciona el horno 2, funciona el horno 3 y falla el horno 4. El sistema funciona ya que el papel sufre una compresión adecuada durante los tres primeros hornos.

Evento 2: Funciona el horno 1, funciona el horno 2, falla el horno 3 y funciona el horno 4. El sistema funciona

ya que la compresión que sufre el papel durante los dos primeros hornos y durante el último es adecuada.

Evento 3: Funciona el horno 1, funciona el horno 2, falla el horno 3 y falla el horno 4. El sistema no funciona ya que la compresión que sufre el papel durante los dos primeros hornos no es suficiente y el paquete tendrá que envolverse de nuevo.

Evento 4: Funciona el horno 1, falla el horno 2, falla el horno 3 y funciona el horno 4. El sistema no funciona ya que la compresión que sufre el papel durante los dos hornos intermedios no es suficiente y el paquete tendrá que envolverse de nuevo.

Sistemas CK/N:F ponderados:

Considere la Fig. 1.5:

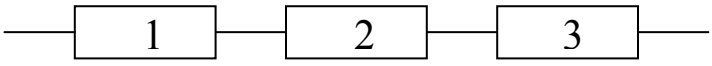


Fig. 1.5 Sistema C3/3:F ponderado

La fig. 1.5 representa el DBC de un sistema de bombeo formado por tres estaciones de bombeo conectadas en serie. La estación 1 tiene 2 bombas, la estación 2 tiene 3 bombas y la estación 3 tiene 2 bombas, las bombas son idénticas, no fallan y están conectadas en paralelo en cada estación.

El ingeniero encargado de la operación de este sistema nos dice que si perdemos al menos 3 bombas debido a fallas de estaciones consecutivas el sistema falla ya que se dispara por baja presión. Supongamos los siguientes eventos:

Evento 1: Falla la estación 1 (perdemos 2 bombas), funciona la estación 2 (contamos con 3 bombas) y falla la estación 3 (perdemos 2 bombas), entonces el sistema funciona ya que estamos perdiendo a lo máximo solo 2 bombas de estaciones consecutivas.

Evento 2: Falla la estación 1 (perdemos 2 bombas), falla la estación 2 (perdemos 3 bombas) y funciona la estación 3 (contamos con 2 bombas), entonces el sistema falla ya que estamos perdiendo 5 bombas de estaciones consecutivas (2 bombas de la estación 1 + 3 bombas de la estación 2).

Sistemas CCK/N:F ponderados:

Considere la Fig. 1.6:

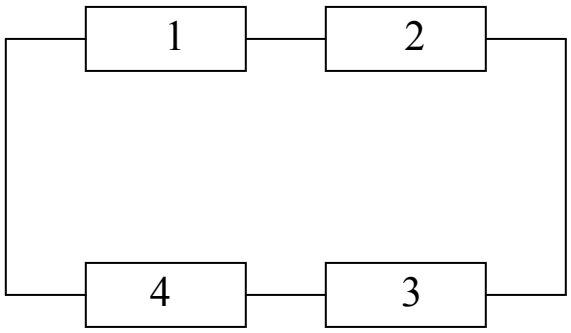


Fig. 1.6 Sistema CC100/4:F ponderado

La fig. 1.6 representa el DBC de una red formada por cuatro computadoras las cuáles constituyen la base de datos distribuida de un programa que requiere una base de datos muy confiable debido al tipo de cálculos que realiza. La cantidad de datos almacenada por terminal es: terminal 1 → 25 datos, terminal 2 → 75 datos, terminal 3 → 25 datos y el terminal 4 → 50 datos.

El diseñador del programa nos dice que si se pierden al menos 100 datos pertenecientes a computadoras consecutivas, el programa no puede realizar el computo deseado y el sistema falla. Supongamos los siguientes eventos:

Evento 1: Funcionan los terminales 1 y 4, fallan los terminales 2 y 3, en este caso el sistema falla ya que estamos perdiendo 100 datos de terminales consecutivos (75 del terminal 2 + 25 del terminal 3).

Evento 2: Funcionan los terminales 2 y 3, fallan los terminales 1 y 4, en este caso el sistema funciona ya que estamos perdiendo sólo 75 datos de terminales consecutivos (25 del terminal 1 + 50 del terminal 4).

Sistemas LCCS

Considere la Fig. 2.5:

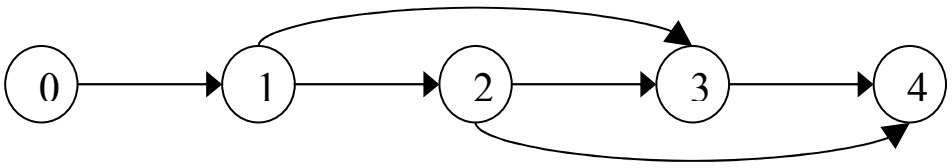


Fig. 2.5 Sistema LCCS de 3 componentes intermedios

La fig. 2.5 representa el DBC de un sistema de telecomunicaciones formado por una estación transmisora (estación 0), tres estaciones repetidoras (estaciones 1, 2 y 3) y una estación receptora (estación 4). El sistema funciona si hay una conexión de la estación transmisora a la estación receptora a través de las estaciones repetidoras. La estación receptora no falla, los enlaces tampoco fallan. Supongamos los siguientes eventos:

Evento 1: Funcionan las estaciones 0, 1, 3 y 4 y falla la estación 2, el sistema funciona ya que existe una conexión entre transmisor y receptor.

Evento 2: Funcionan las estaciones 0, 2, 3 y 4 y falla la estación 1, el sistema no funciona ya que no existe una conexión entre transmisor y receptor.

Evento 3: Funcionan las estaciones 0, 1, 2 y 4 y falla la estación 3, el sistema funciona ya que existe una conexión entre transmisor y receptor.