Centro de Ciencias Básicas

Ciencias de la Computación, Estadística, Matemáticas y Física

TESIS

OPTIMIZACION DE SISTEMAS MULTIESTADO BASADO EN LA CONFIABILIDAD

PRESENTA

Brenda Estefanía Morales Calzada

PARA OBTENER EL GRADO DE MAESTRA EN CIENCIAS CON OPCION A LA COMPUTACION

Tutores

Dr. Ángel Eduardo Muñoz Zavala

Dr. Jorge Eduardo Macias Diaz.

Asesor

Dr. José Antonio Guerrero Diaz de León.

Aguascalientes, Ags, 29 de Noviembre del 2023.

CARTA DE VOTO APROBATORIO
COMITÉ TUTORAL

MTRO.JORGE MARTIN ALFÉREZ CHÁVEZ
DECANO (A) DEL CENTRO DE CIENCIAS BASICAS

P R E S E N T E

Por medio del presente como **Miembros del Comité Tutoral** designado del estudiante **BRENDA ESTEANIA MORALES CALZDA** con ID **137973** quien realizó la tesis titulado: **OPTIMIZACIÓN DE SISTEMAS MULTIESTADO BASADO EN LA CONFIABILIDAD** , un trabajo propio, innovador, relevante e inédito y con fundamento en el Artículo 175, Apartado II del Reglamento General de Docencia damos nuestro consentimiento de que la versión final del documento ha sido revisada y las correcciones se han incorporado apropiadamente, por lo que nos permitimos emitir el **VOTO APROBATORIO**, para que ella pueda proceder a imprimirla así como continuar con el procedimiento administrativo para la obtención del grado.

Ponemos lo anterior a su digna consideración y sin otro particular por el momento, le enviamos un cordial saludo.

A T E N T A M E N T E
"Se Lumen Proferre"
Aguascalientes, Ags., a 29 de Noviembre de 2023.

Dr. Ángel Eduardo Muñoz Zavala.
Co-Tutor de tesis

Dr. Jorge Eduardo Macias Diaz.
Co-Tutor de tesis

Dr. José Antonio Guerrero Diaz de León.
Asesor de tesis

c.c.p.- Interesado
c.c.p.- Secretaría Técnica del Programa de Posgrado

DICTAMEN DE LIBERACION ACADEMICA PARA INICIAR LOS TRAMITES DEL EXAMEN DE GRADO

Fecha de dictaminación dd/mm/aaaa: 29/11/2023

NOMBRE: Brenda Estefania Morales Calzada **ID** 137973

PROGRAMA: MAESTRIA EN CIENCIAS CON OPCION A LA COMPUTACION **LGAC (del posgrado):** Inteligencia Artificial

TIPO DE TRABAJO: (X) Tesis ( ) Trabajo Práctico

TITULO: Optimización de Sistemas multiestado basado en la confiabilidad

IMPACTO SOCIAL (señalar el impacto logrado): La tesis aporta una intergración de la Función Generadora Universal, utilizada para evaluar la confibilidad de sistemas, en conjunto con la técnica de optimización de Colonia de Hormigas; cuyo impacto es minimizar el costo en el diseño de sistemas sujetos a proabilidades de vida.

INDICAR SI NO N.A. (NO APLICA) SEGÚN CORRESPONDA:

	Elementos para la revisión académica del trabajo de tesis o trabajo práctico:
SI	El trabajo es congruente con las LGAC del programa de posgrado
SI	La problemática fue abordada desde un enfoque multidisciplinario
SI	Existe coherencia, continuidad y orden lógico del tema central con cada apartado
SI	Los resultados del trabajo dan respuesta a las preguntas de investigación o a la problemática que aborda
SI	Los resultados presentados en el trabajo son de gran relevancia científica, tecnologíca o profesional según el área
SI	El trabajo demuestra más de una aportación original al conocimiento de su área
SI	Las aportaciones responden a los problemas prioritarios del país
NO	Generó transferecia del conocimiento o tecnológica
SI	Cumple con la ética para la investigación (reporte de la herramienta antiplagio)
	El egresado cumple con lo siguiente:
SI	Cumple con lo señalado por el Reglamento General de Docencia
SI	Cumple con los requisitos señalados en el plan de estudios (créditos curriculares, optativos, actividades complementarias,estancia, predoctoral, etc)
SI	Cuenta con los votos aprobatorios del comité tutoral, en caso de los posgrados profesionales si tiene solo tutorpodrá liberar solo el tutor
N.A.	Cuenta con la carta de satisfacción del Usuario
SI	Coincide con el título y objetivo registrado
SI	Tiene congruencia con cuerpos académicos
SI	Tiene el CVU del Conacyt actualizado
N.A.	Tiene el artículo aceptado o publicado y cumple con los requisitos institucionales (en caso que proceda)
	En caso de Tesis por artículos científicos publicados
N.A.	Aceptación o Publicación de los articulos según el nivel del programa
N.A.	El estudiante es el primer autor
N.A.	El autor de correspondencia es el Tutor del Núcleo Académico Básico
N.A.	En los artículos se ven reflejados los objetivos de la tesis, ya que son producto de este trabajo de investigación.
N.A.	Los artículos integran los capítulos de la tesis y se presentan en el idioma en que fueron publicados
N.A.	La aceptación o publicación de los artículos en revistas indexadas de alto impacto

Con base a estos criterios, se autoriza se continúen con los trámites de titulación y programación del examen de grado:

Sí X

No

FIRMAS

Elaboró:

* NOMBRE Y FIRMA DEL CONSEJERO SEGÚN LA LGAC DE ADSCRIPCION: Dr. Hermilo Sánchez Cruz

NOMBRE Y FIRMA DEL SECRETARIO TÉCNICO: Dr. Hermilo Sánchez Cruz

*** En caso de conflicto de intereses, firmará un revisor miembro del NAB de la LGAC correspondiente distinto al tutor o miembro del comité tutoral, asignado por el Decano**

Revisó:

NOMBRE Y FIRMA DEL SECRETARIO DE INVESTIGACIÓN Y POSGRADO: P.A. Dr. Juan Jáuregui Rincón.

Autorizó:

NOMBRE Y FIRMA DEL DECANO: M. en C. Jorge Martín Alférez Chávez.

Nota: procede el trámite para el Depto. de Apoyo al Posgrado

En cumplimiento con el Art. 105C del Reglamento General de Docencia que a la letra señala entre las funciones del Consejo Académico: .... Cuidar la eficiencia terminal del programa de posgrado y el Art. 105F las funciones del Secretario Técnico, llevar el seguimiento de los alumnos.

Agradecimientos

A la Benemérita Universidad Autónoma de Aguascalientes por ser mi casa formadora desde 2013.

Al CONAHCYT por la confianza dada al otorgarme una beca de investigador en lo que duro mi maestría.

A los doctores, José Eduardo Macias Días y José Antonio Diaz de León por su apoyo como co-tutor y parte del comité tutoral respectivamente, pero de un modo especial al Dr. Ángel Eduardo Muñoz Zavala por siempre confiar en mí, y creer más de lo que yo creía en mí.

Dedicatorias

A Dios que es el centro de mi vida.

A mis padres por el apoyo incondicional, y nunca nos dejan derrotarnos.

A mis hermanos, mis mejores amigos y siempre presentes en cada etapa.

A mi tía Claudia Calzada que es una segunda madre y ejemplo siempre fuerza.

A todos los que directa o indirectamente ayudaron a terminar lo que inicie.

índice General

Índice de tablas

Índice de Ilustraciones

Acrónimos

ACO	Algoritmo de colonia de hormigas.
GA	Algoritmo Genético.
UGF	Función Generadora Universal.
MSS	Sistema Multi-estado.
SMS	Sistema Multi Estado.

Resumen

El siguiente trabajo es un diseño que habla sobre la unión de un sistema de colonia de hormigas como algoritmo de optimización y de la función universal para encontrar un punto donde la disponibilidad sea alta y tenga un costo bajo, todo esto programado en Rstudio, que es un software libre.

Para lo cual la unión del evaluador y el optimizador se realizaron varias corridas para encontrar si era factible el uso de el ACO y la unión del optimizador, teniendo en cuenta que este problema, no se necesita la rapidez de la solución, si no que sea una solución que cumpla los puntos que se piden en el libro, que sea de bajo costo y de una disponibilidad especifica, que se puede cambiar, el tope de la disponibilidad o el de el costo.

Abstract

The following work is a design study about the union of an ant colony system as an optimization algorithm and the universal function to find a point where availability is high and has a low cost, all of this programmed in Rstudio, which is a free software.

For which the union of the evaluator and the optimizer, several runs were carried out to find if the use of the ACO and the union of the optimizer were feasible, taking into account this problem, the speed of the solution isn´t needed, but it must be a solution that meets the points requested in the book, that is low cost and has a specific availability, which can be changed, the availability limit or the cost limit.

1. Introducción

1.1 Preámbulo

Existen sistemas que logran desempeñar una tarea día a día, pero estos se han vuelto cada vez más complejos, ya que estos sistemas van desde los que pueden realizar algún proceso en la industria que da como resultado algún producto final y los más complejos que necesitan de pequeños sistemas para crear uno de mayor tamaño, como puede ser un componente electrónico que forman parte de un aparato más sofisticado, compuesto de componentes más pequeños. Los sistemas que constan de una serie de componentes o elementos que están conectados entre sí para lograr la generación de un sistema más complejo.

Debido a los diferentes estados que contiene el producto final, cada uno puede funcionar de manera diferente, y la suma del funcionamiento de estos estados se le conocer como Niveles de desempeño del sistema. (Anatoly Lisnianski et al., 2003), ya que se tiene diferente niveles de desempeño en cada sistema, la confiabilidad que según la RAE es la probabilidad de un buen funcionamiento, si se tiene una confiabilidad superior de cada subsistema ,este puede aumentar los costos en la industria y como consecuente elevados costos en la venta al público en general, pero es aquí donde se puede utilizar un problema de optimización , ya se maximizando la confiabilidad de nuestro sistema final y minimizando los costos para así aprovechar el sistema lo más posible.

Que es lo que abordaremos en esta tesis de investigación donde con la ayuda de la metaheurística para problemas de optimización, los algoritmos Genéticos en especial el Algoritmo por colonia de hormigas que esta basado en la naturaleza y los procesos que llevan acabo las hormigas ya sea en la comunicación para búsqueda de su alimento , con base en esto, se realizara un algoritmo que nos ayude a maximizar la confiabilidad de un sistema, basándonos en el problema de optimización que proponen (Anatoly Lisnianski et al., 2003), en su libro. Multi-state System Reliability: Assessment, Optimization and applications.

1.2 Justificación

El conocimiento de Algoritmos evolutivos que ayudan a la industria para mejorar sus tiempos de construcción de algún producto, y a la vez siempre basados en la confiabilidad y bajo costo, que al final es lo mas importante para el usuario e indirectamente es esencial para las empresas, para que sus productos sigan comercializándose, pero sin tener perdidas de coste en el camino.

1.3 Objetivo

Proponer una metodología de inteligencia artificial para maximizar la confiabilidad del sistema, basado en el tipo y cantidad de componentes multi-estado pertinentes para el funcionamiento de dicho sistema.

1.4 Hipótesis

La utilización de la Función Generadora Universal en conjunto con los algoritmos evolutivos puede minimizar el costo en el diseño de diversos tipos de sistemas multi-estado sujetos a una probabilidad de vida (confiabilidad) deseada.

1.5 Planteamiento del Problema de Investigación

Encontrar una metodología de inteligencia artificial que maximice la confiabilidad de los sistemas, basado en el tipo y cantidad de componentes multi-estado pertinentes para el funcionamiento de dicho sistema.

2. Referentes teóricos y contextuales

2.1 Sistema Multi-estado.

Los sistemas que logran desempeñar una tarea día a día, pero estos se han vuelto cada vez más complejos, ya que estos sistemas van desde los que pueden realizar algún proceso en la industria que da como resultado algún producto final y los más complejos que necesitan de pequeños sistemas para crear uno de mayor tamaño, como puede ser un componente electrónico que forman parte de un aparato más sofisticado, compuesto de componentes más pequeños. Los sistemas que constan de una serie de componentes o elementos que están conectados entre sí para lograr la generación de un sistema más complejo. (Anatoly Lisnianski et al., 2003)

A estos diferentes sistemas que componen un solo elemento final se le llama Sistema Multi-estado. Un sistema que puede tener un número finito de tasas de rendimiento y que se compone por elementos que a su vez pueden ser multi-estado. La tasa de rendimiento de ese sistema está dado por diferentes unidades que tienen un efecto acumulativo del rendimiento del sistema, y esa tasa de rendimiento depende de la disponibilidad de sus unidades, ya que cada unidad da como resultado un nivel de desempeño distinto para cada una de estas unidades. (Anatoly Lisnianski et al., 2003)

La tasa de rendimiento de los elementos puede variar o tener fallas, estas fallas conducen a una disminución de rendimiento por elementos del sistema y a esto se le conoce como falla parcial del sistema, después de una falla parcial los demás elementos pueden seguir funcionando, pero con tasa de rendimiento reducidas, y así poco a poco llegar a una falla completa donde estos sistemas son incapaces de seguir realizando sus tareas. (Anatoly Lisnianski et al., 2003)

Desde la década de 1960 se han propuesto y desarrollado muchos modelos diversificados y métodos de solución, donde se utiliza la confiabilidad como medida del desempeño del sistema. (Anatoly Lisnianski et al., 2003)

2.2 Redundancia

Se obtiene una nueva configuración óptima del sistema cuando en el diseño intervienen redundancias activas y de espera en frío. Un componente redundante en espera en frío no falla antes de ponerse en funcionamiento mediante la acción de conmutación, mientras que el patrón de fallo de un componente redundante activo no depende de si el componente está inactivo o en funcionamiento. La redundancia en espera en frío puede proporcionar una mayor confiabilidad, pero es difícil de implementar debido a las dificultades involucradas en la detección y conmutación de fallas.

2.3 Confiabilidad

La idea intuitiva sobre la confiabilidad de un equipo o sistema, de cualquier naturaleza, se relaciona con su habilidad o capacidad de realizar una tarea específica. Por esta razón, normalmente es considerada una propiedad cualitativa más que cuantitativa. Sin embargo, se debe convenir en que, para la práctica de ingeniera, resulta mucho más atractivo disponer de un índice cuantitativo, especialmente cuando se desea tomar una decisión sobre alternativas de diseño que cumplen finalmente las mismas funciones. Esta cuantificación de la habilidad de un sistema, se denomina confiabilidad, o bien fiabilidad y puede expresarse por una gran variedad de índices, dependiendo de los objetos que se persigan con la evaluación.

La disponibilidad, puede ser definida como la confianza de que un componente o sistema ejerza su función satisfactoriamente para un tiempo dado. En la práctica, la disponibilidad se expresa como el porcentaje de tiempo en que el sistema está listo para operar o producir, esto en sistemas que operan continuamente.

En la fase de diseño de equipos o sistemas, se debe buscar el equilibrio entre la disponibilidad y el costo. Dependiendo de la naturaleza de requisitos del sistema, el diseñador puede alterar los niveles de disponibilidad, confiabilidad y mantenibilidad, de forma a disminuir el costo total del ciclo de vida.

Existen muchos algoritmos para evaluar la confiabilidad de diferentes tipos de sistemas binarios. (Grajales et al., 2006)

Dado que se desarrollaron algoritmos especializados muy eficaces para cada tipo de sistema, los procedimientos basados en UGF pueden no parecer muy eficaces en comparación con los algoritmos más conocidos.

El análisis de confiabilidad del sistema considera la relación entre el funcionamiento de los elementos del sistema y el funcionamiento del sistema en su conjunto. Un elemento es una entidad en un sistema que no se subdivide más. Esto no implica que un elemento no pueda estar formado por partes; más bien significa que, en un estudio de confiabilidad determinado, se lo considera una unidad autónoma y no se analiza en términos del funcionamiento de sus componentes.

En el análisis de confiabilidad del sistema binario se supone que cada elemento del sistema, así como el sistema completo, puede estar en uno de dos estados posibles, es decir, funcionando o fallando. Por lo tanto, el estado de cada elemento o del sistema se puede representar mediante una variable aleatoria binaria tal que X_j indica el estado del elemento j: X_j=1, si el elemento j está en condiciones de funcionamiento y X_j=0, si el elemento j está fallado; X, indica el estado de todo el sistema: X = 1, si el sistema funciona, X = 0, si el sistema falla.

Los estados de los n elementos que componen el sistema están representados por el llamado vector de estado de los elementos (X_1,…, X_n). Se supone que los estados de los elementos del sistema (la realización del vector de estado de los elementos) determinan inequívocamente el estado del sistema. Por tanto, la relación entre el vector de estado del elemento y la variable de estado del sistema X puede expresarse mediante la función determinista.

$$X = \phi(X_1, \dots, X_n)$$

Esta función se llama función de estructura del sistema.(Levitin & Cai, 2011),cap.2,pp.29-30.

2.4 Técnicas de evaluación

Se utiliza la Función Generadora Universal (UGF) para evaluar la disponibilidad de un sistema multi-estado en serie-paralelo con recursos computacionales pequeños, donde la propiedad esencial de la transformada U permite obtener la función U total para un MSS con componentes conectados en paralelo o serie , mediante operaciones algebraicas simples que involucran funciones U de componentes individuales.

En la literatura, se ha desarrollado y aplicado ampliamente una importante estrategia de optimización, que combina Función Generadora Universal (UGF) y Algoritmos Genéticos (GA), a los problemas de optimización de la confiabilidad de los Sistemas multi-estado (MSS), es importante encontrar una técnica de codificación y decodificación eficaz para mejorar la eficiencia del GA. (Levitin G, 2005).

2.5 Técnicas de optimización

Debido a la robustez y viabilidad, los algoritmos metaheurísticos, específicamente los Algoritmos Genéticos, sé han aplicado amplia y exitosamente para mejorar la eficiencia del cálculo o evitar convergencia prematura, o ya sea otros métodos de búsqueda local. Aunque implica un mayor esfuerzo de cálculo, los métodos exactos son particularmente ventajosos para problemas pequeños y sus soluciones pueden usarse para medir el desempeño de los métodos heurísticos o metaheurísticos. (Levitin G, 2005).

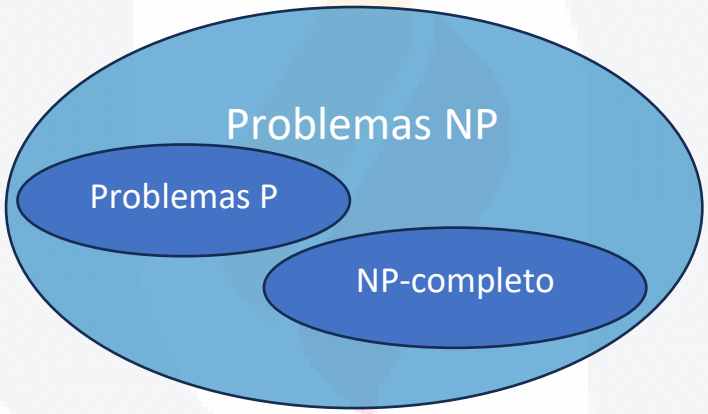
2.6 Métodos Metaheurísticos

Las metaheurísticas constituyen uno de los conjuntos de las metodologías en el trabajo y la investigación con fenómenos, sistemas y comportamientos caracterizados por no linealidad, autoorganización, complejidad creciente, emergencia y otras propiedades semejantes y conocidas.

Las metaheurísticas operan mediante algoritmos, salvo que no son algoritmos de orden común, son especiales, básicamente porque no se rigen por un patrón predictivo, ni causal, ni organizado, más bien es aleatorio: el algoritmo adquiere su “forma óptima” a través de itinerancias o pruebas que aproxima la solución; por ejemplo, mediante métodos de búsqueda basados en gradientes. Los algoritmos más conocidos en metaheurística son los algoritmos genéticos, la búsqueda tabú, ACO (algoritmo de colonia de hormigas, por sus siglas en ingles), recocido simulado (simulated annealing), PSO (partículas de optimización de enjambre, por sus siglas en ingles), y en forma común sirven para el mismo propósito general: tratar de recorrer el espacio de soluciones sin quedar “atrapados” en una zona.

Si bien la forma de ver la complejidad de estos problemas es mediante la clasificación de estas clases, que son problemas tipo P versus NP. La Ilustración 1 representa esta clasificación.(Goyal et al., 2021)

ILUSTRACIÓN 1 :CLASE DE PROBLEMAS PY NP.



3. Metodología

En este capítulo se explicarán las metodologías con respecto al Algoritmo evolutivo y la utilización de la función generadora universal explicada de manera matemática, que es lo que se utilizo para el programa que se presenta en esta investigación.

3.1 Optimización basada en colonia de Hormigas (ACO)

3.1.1 Definición

Las hormigas son insectos sociales que viven en colonias y que, debido a su colaboración, son capaces de mostrar comportamientos complejos y realizar tareas difíciles desde el punto de vista de una hormiga individual. Un aspecto interesante del comportamiento de muchas especies de hormigas es su habilidad para encontrar los caminos más cortos entre su hormiguero y las fuentes de alimento. Este hecho es especialmente interesante si se tiene en cuenta que muchas de las especies de hormigas son casi ciegas, lo que evita el uso de pistas visuales.(Alonso et al., 2004)

Mientras que se mueven entre el hormiguero y la fuente de alimento, algunas especies de hormigas depositan una sustancia química denominada *feromona* (una sustancia que puede “olerse”). Si no se encuentra ningún rastro de feromona, las hormigas se mueven de manera básicamente aleatoria, pero cuando existe feromona depositada, tienen mayor tendencia a seguir el rastro.(Alonso et al., 2004)

En la práctica, la elección entre distintos caminos toma lugar cuando varios caminos se cruzan. Entonces, las hormigas eligen el camino a seguir con una decisión probabilística sesgada por la cantidad de feromona: cuanto más fuerte es el rastro de feromona, mayor es la probabilidad de elegirlo. Puesto que las hormigas depositan feromona en el camino que siguen, este comportamiento lleva a un proceso de autorefuerzo que concluye con la formación de rastros señalados por una concentración de feromona elevada. Este comportamiento permite además a las hormigas encontrar los caminos más cortos entre su hormiguero y la fuente del alimento.(Alonso et al., 2004)

3.1.2 Metodología del ACO

Dado que se trabajara un problema de combinatoria, donde se requiere una minimización ya que la función objetivo será representada con los costos, donde el algoritmo puede enfrentar un impacto de construcción, que con las condiciones propuestas en el problema del que se guía ("The Universal Generating Function in Reliability Analysis and Optimization", 2005), donde Gutjahr, W.J. (2000) define los conceptos de un modelo matemático para la utilización de este algoritmo ,donde los elementos para aplicarlo son los siguientes:

1. Documento que representa todos los nodos posibles a visitar con su probabilidad de elección.
2. Un conjunto $A_1, A_2, \dots, A_s$,de agentes ("hormigas") donde cada uno construye una caminata aleatoria con ciertas restricciones eligiendo los nodos a visitar en base a una matriz que contiene probabilidades de transición. El nodo $i \to j$, representa el ir del punto i al j. Cada una de las caminatas o caminos de las hormigas es construido separadamente y de modo secuencial. El periodo de tiempo en el que una hormiga construye un camino es llamado ciclo y definimos L_k como la distancia total recorrida en el camino de la hormiga k.
3. Las probabilidades de transición de los movimientos aleatorios de las hormigas dentro de cada ciclo se encuentran en una matriz P que es de transición.

   Sea $u = (u_0, \dots, u_{t-1})$ el camino parcial construido por una hormiga antes del $t - ésimo$ paso en un ciclo m. Así, u_0 indica el nodo inicial, u_1 el siguiente nodo a u_0, etc. Decimos que l _ u si l está contenido en u: Si A es el conjunto de arcos que une los nodos en la grá.ca de construcción, entonces la forma general de las probabilidades de transición están dadas por:

$$p_{kl}(m,u) = \begin{cases} \dfrac{[\mathcal{T}_{kl}(u)]^{\alpha}[\eta_{kl}(u)]^{\beta}}{\left(\sum_{(k,r)}[\mathcal{T}_{kr}(u)]^{\alpha}\left[\eta_{k,r}(u)\right]^{\beta}\right)}; & si\ l \in u \\ 0; & si\ l \notin u \end{cases}$$

   Donde $p_{kl}(m,u)$ muestra la probabilidad que tiene una hormiga para moverse al nodo siguiente.

Y $\alpha, \beta > 0$ denotan parámetros que asignan una toma de decisión, tomando en cuanta la cantidad que se deposita por feromona, al inicio del ciclo la hormiga se coloca de manera aleatoria en un nodo para comenzar ahí su nuevo camino.

4. Las Feromonas $\mathcal{T}_{kl}$ construyen una matriz $\mathcal{T}$, donde cada posición corresponde a que cantidad de veces se eligió ese nodo, para que alguna otra hormiga tome la decisión de pasar o no por ahí, que es donde se presenta la actualización de las feromonas, y se lleva acabo de la siguiente manera:
    - Evaporación: A cada valor de la matriz de feromonas se aplica esta actualización que esta dado por $\mathcal{T}_{kl} = (1-\rho)\mathcal{T}_{kl}$. Donde ρ es la evaporación, y $(1-\rho)$ es el porcentaje de feromonas que quedan en esa elección.
    - Deposito: una vez realizada la evaporación, se realizará el depósito en cada uno de los puntos que se dejo la feromona, y se hace con la siguiente formula; $\mathcal{T}_{kl} = Q/L_k$,donde Q es el parámetro de aprendizaje que deja la feromona, y L_k es el costo total que lleva el camino o sistema que recorrió, en el caso que no se halla recorrido, solo se coloca un cero.

Una vez terminado el deposito de la feromona en el camino total que todas las hormigas recorren para encontrar un óptimo, se genera una nueva matriz de probabilidades que es la que la siguiente generación utilizara para comenzar, y a partir de estas probabilidades que se irán actualizando en cada generación, encontrara una serie de caminos óptimos para poder recorrer.

Estas son las condiciones que aplica el ACO para realizar una serie de procesos estocásticos, para dar pie a probabilidades de soluciones en un problema de optimización con combinatoria, donde se puede observar que lo que llamamos hormigas, son las que buscan el mejor camino o el camino optimo de combinatoria donde con la ayuda de la feromona eligen tal camino.

3.2 Función Generadora Universal

En los últimos años se ha aplicado ampliamente un enfoque específico llamado técnica de función generadora universal (UGF) al análisis de confiabilidad de MSS. La técnica UGF permite encontrar la distribución de rendimiento completa de MSS en función de las distribuciones de rendimiento de sus elementos utilizando procedimientos algebraicos. Esta técnica (a veces también llamada método de secuencias generadoras generalizadas)

(Gnedenko y Ushakov 1996) generaliza la técnica que se basa en una función generadora ordinaria bien conocida. Las ideas básicas del método fueron introducidas principalmente por I. Ushakov a mediados de los años 1980 (Ushakov 1986, 1987). Luego, el método fue descrito en un libro de Reinshke y Ushakov (1988), donde se dedicó un capítulo a la UGF. (Desafortunadamente, este libro se publicó sólo en alemán y ruso, por lo que permaneció desconocido para los angloparlantes). La amplia aplicación del método al análisis de confiabilidad de MSS comenzó a mediados de la década de 1990, cuando se informó la primera aplicación (Lisnianski et al. 1994). y se publicaron dos artículos correspondientes (Lisnianski et al. 1996; Levitin et al. 1998). Desde entonces, el método se ha ampliado considerablemente en numerosos trabajos de investigación y en los libros de Lisnianski y Levitin (2003) y Levitin (2005).

En la literatura, se ha desarrollado y aplicado ampliamente una importante estrategia de optimización, que combina UGF y GA, para la optimización de la confiabilidad. Problemas de los SMS renovables. En esta estrategia, hay dos tareas principales: de acuerdo con la estructura del sistema y la naturaleza física del sistema, obtener el UGF del sistema a partir de los UGF componentes; Encuentre una técnica de codificación y decodificación eficaz para mejorar la eficiencia del GA.

Se utiliza primero un enfoque UGF para evaluar la disponibilidad de un sistema multi-estado paralelo en serie con recursos computacionales relativamente pequeños. La propiedad esencial de la transformada U permite obtener la función U total para un MSS con componentes conectados en paralelo o en serie mediante operaciones algebraicas simples que involucran funciones U de componentes individuales. El operador Ω_ω está definido por (1) - (3).

$$\Omega_\omega\big(U_1(z),U_2(z)\big)=\Omega_\omega\left[\sum_{i=1}^{I}p_{1i}p_{2j}z^{g_{1i}},\sum_{j=1}^{j}p_{2j}z^{g_{2j}}\right]=\sum_{i=1}^{I}\sum_{j=1}^{j}p_{1i}p_{2j}z^{\omega(g_{1i},g_{2j})}$$

$$\Omega_\omega\big(U_1(z),\dots,U_k(z),U_{k+1}(z),\dots\ U_n(z)\big)=\Omega_\omega\big(U_1(z),\dots,U_{k+1}(z),U_k(z),\dots\ U_n(z)\big)$$

$$\Omega_\omega\big(U_1(z),\dots,U_k(z),U_{k+1}(z),\dots\ U_n(z)\big)=\Omega_\omega\left(\Omega_\omega\big(U_1(z),\dots,U_k(z)\big),\Omega_\omega\big(U_{k+1}(z),\dots\ U_n(z)\big)\right)$$

Donde la función $\omega(.)$ toma la forma de

$$\omega_{s1}(g_1, g_2) = \min(g_1, g_2)$$

$$\omega_{p1}(g_1, g_2) = \ g_1 + g_2$$

Ya que el enfoque que se le da con la función generadora universa con un sistema multi-estado, se genera con elementos que son dependientes.

Con esta estrategia la función generadora universal y los algoritmos genéticos resuelven la optimización de la estructura de un sistema multi-estado con redundancia.

Donde la estrategia de aplicar estos sistemas consta de dos partes:

-RGS que incluye una serie de subsistemas generadores.

-MPS que incluye elementos que consumen una cantidad fija de recursos para la realización de sus tareas.

El rendimiento total del sistema depende del estado de cada subsistema en el RGS y de la máxima productividad posible del MPS, que esta dada por un problema de programación lineal relacionado con estados del RGS.

Donde se presentan dos diseños óptimos para maximizar la disponibilidad del sistema a los niveles deseados para ambos casos, se muestran a continuación.

Tipo 1.

$$\omega_s{}^O(g_1, g_2) = \omega_s{}^C(g_1, g_2) = \min(g_1, g_2)$$

$$\omega_p{}^O(g_1, g_2) = \omega_p{}^C(g_1, g_2) = g_1 + g_2$$

$$F_C(G_C, W_C) = G_C - W_C \geq 0$$

$$F_O(G_O, W_O) = G_O - W_O \geq 0$$

Tipo 2.

$$\omega_s{}^O(g_1, g_2) = \min(g_1, g_2)$$

$$\omega_p{}^O(g_1, g_2) = \max(g_1, g_2)$$

$$\omega_s{}^C(g_1, g_2) = \max(g_1, g_2)$$

$$\omega_p{}^{C}(g_1, g_2) = \min(g_1, g_2)$$

$$F_C(G_C, W_C) = W_C - G_C \geq 0$$

$$F_O(G_O, W_O) = W_O - G_O \geq 0$$

Por lo tanto, la disponibilidad del sistema se puede denotar por:

$$A_s(t) = 1 - \Pr\{F_c(G_c(t), W_c) < 0\} - \Pr\{F_O(G_O(t), W_O) < 0\}$$

Posteriormente introduce un parámetro de probabilidad de 0,5 para ambos modos e incluso extiende esta técnica para evaluar la disponibilidad de sistemas con estructuras de puentes.(Levitin & Cai, 2011).p.12-16.

3.2.1 Función Generadora Universal en la optimización de sistemas multi-estado en serie-paralelo

La mejora de la confiabilidad es de importancia crítica en varios tipos de sistemas; sin embargo, cualquier esfuerzo para este tipo de mejora generalmente requiere recursos que están limitados por técnicas o recursos económicos. Se pueden distinguir dos enfoques en el problema de optimización de la confiabilidad. El primero tiene como objetivo lograr la mayor confiabilidad posible sujeto a diferentes restricciones, este problema se denomina problema de optimización de confiabilidad directa, y el segundo se enfoca en minimizar los recursos necesarios para proporcionar un rendimiento requerido, nivel de confiabilidad.

Existen cuatro métodos generales para mejorar la confiabilidad del sistema:

- Una disposición de despido;

- Un ajuste óptimo de los parámetros del sistema, una disposición óptima de los elementos existentes o la asignación de elementos intercambiables;

- Una mejora de la confiabilidad (disponibilidad) y/o del rendimiento de los elementos del sistema;

- Una combinación de los métodos mencionados anteriormente.

Aplicados a un MSS, estos métodos afectan dos propiedades básicas del sistema: su configuración (función estructural) y la distribución del rendimiento de sus elementos.

El método UGF que permite evaluar la distribución del rendimiento del sistema y, por lo tanto, evaluar sus medidas de rendimiento basándose en un procedimiento rápido, abre nuevas posibilidades para resolver problemas de optimización de la confiabilidad de MSS. Con base en la técnica UGF, la confiabilidad del MSS se puede obtener en función de la estructura del sistema y las distribuciones de desempeño de sus elementos. Por lo tanto, se pueden formular numerosos problemas de optimización en los que la composición óptima de todos o parte de los factores que influyen en la confiabilidad total del MSS debe encontrarse sujeta a diferentes restricciones.

Teniendo la estructura del sistema definida por el diagrama de bloques de confiabilidad de sus componentes y por el conjunto $\{\boldsymbol{n}_1, \boldsymbol{n}_2, \dots, \boldsymbol{n}_N\}$, se puede determinar el MSS completo.

La medida de desempeño $O(\boldsymbol{w}, \boldsymbol{q}, \boldsymbol{n}_1, \boldsymbol{n}_2, \dots, \boldsymbol{n}_N)$ para cualquier distribución de demanda dada **w, q**. El problema de optimización de la estructura MSS se formula como encontrar la configuración del sistema de costo mínimo $\{\boldsymbol{n}_1, \boldsymbol{n}_2, \dots, \boldsymbol{n}_N\}$ que proporcione el nivel requerido O* de la medida de desempeño del sistema O:

$$C(\boldsymbol{n}_1, \boldsymbol{n}_2 \dots, \boldsymbol{n}_N) \rightarrow minimo\ sujeto\ a\ f(O, O^*) = 1$$

(“The Universal Generating Function in Reliability Analysis and Optimization”, 2005), p.191.

3.3 Caso de Estudio

En el ejemplo 5.1 (“The Universal Generating Function in Reliability Analysis and Optimization”, 2005)que en particular donde está basado nuestro sistema para la realización de el código, muestra que es un sistema de transporte de carbón de una central eléctrica que conta de cinco componentes básicos conectados en serie como muestra la Ilustración 2.

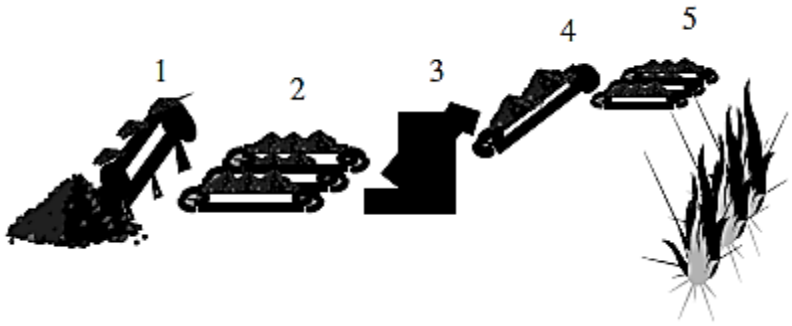


ILUSTRACIÓN 2 SISTEMA EN SERIE DE TRANSMISIÓN DE FLUJO.

1.Subsistema de alimentadores primarios.

2.Subsistema de transportadores primarios.

3.Subsistema de apiladores-recuperadores.

4.Subsistema de alimentadores secundarios.

5.Subsistema de transportadores secundarios.

Donde el sistema pertenece a una transmisión de flujo MSS con dispersión de flujo, que principalmente su característica es la capacidad de transmisión y de elementos paralelos, que estos pueden transmitir el carbón simultáneamente, donde el sistema a satisfacer tiene una demanda variable W y su función de aceptabilidad se define como $F(G,W) = 1(G\ W)$,el sistema debe tener una disponibilidad no menor a A*=0.99 para la distribución de demanda dada w, q presentada en la siguiente Tabla 1.

Tabla 1. Distribución de demanda.

w	1.00	0.80	0.50	0.20
q	0.48	0.09	0.14	0.29

Donde se da a conocer que cada elemento del sistema es un elemento con falla total (lo que implica que solo pude tener dos estados, funcionando o en falla total, correspondiente a una capacidad de cero. Para cada tipo de equipo existe una lista de productos disponibles en el mercado, cada versión de equipo se caracteriza por su capacidad nominal g, disponibilidad p y costo c. La Tabla 2 muestra la lista de la que se está hablando.

Tabla 2. Parámetros de elementos MSS disponibles.

No. de versión del elemento MSS	Componente1			Componente 2			Componente 3			Componente 4			Componente 5		
	Alimentadores primarios			Transportadores primarios			Apiladores-recuperadores			Alimentadores secundarios			Transportadores secundarios		
	g	p	c	g	p	c	g	p	c	g	p	c	g	p	c
1	1.20	0.980	0.590	1.00	0.995	0.205	1.00	0.971	7.525	1.15	0.977	0.180	1.28	0.984	0.986
2	1.00	0.977	0.535	0.92	0.996	0.189	0.60	0.973	4.720	1.00	0.978	0.160	1.00	0.983	0.825
3	0.85	0.982	0.470	0.53	0.997	0.091	0.40	0.971	3.590	0.91	0.978	0.150	0.60	0.987	0.490
4	0.85	0.978	0.420	0.28	0.997	0.056	0.20	0.976	2.420	0.72	0.983	0.121	0.51	0.981	0.475
5	0.48	0.983	0.400	0.21	0.998	0.042				0.72	0.981	0.102			
6	0.31	0.920	0.180							0.72	0.971	0.096			
7	0.26	0.984	0.220							0.55	0.983	0.071			
8										0.25	0.982	0.049			
9										0.25	0.97	0.044			

Donde la definición correcta que se va a utilizar para el caso que se pueden elegir diferentes versiones y número de elementos para cualquier componente del sistema, donde la estructura del componente i del sistema esta definida por los números de los elementos en paralelo de cada versión b elegida para ese componente: $\eta_{ib}(1 \leq b \leq B_i)$.

Donde en el libro muestra que se llegó a la mejor solución de $(0,0,0,2,0,1,0,0,0,2,0,0,0,2,1,0,0,0,0,0,0,0,3,0,0,0,0,0,3)$ con un costo de 15.870 y $A^* = 0.99$.

4. Implementación y Resultados

4.1 Función Objetivo

En este siguiente apartado se mostrará como se realizó la implementación y una pequeña explicación del código utilizado, y que es lo que se muestra en cada etapa. Así como se mostrará los cambios que se realizaron al ACO para poder aplicarlo al caso de estudio que se presenta.

Se desea encontrar el número de componentes en paralelo (redundancia) que se deben elegir de un conjunto de alternativas (proveedores) que minimice los costos del sistema cumpliendo una disponibilidad (confiabilidad) requerida.

4.2 Código ACO Adaptado a Sistemas Multiestado

Se comienza por la generación inicial de la matriz de hormigas, colocando de manera aleatoria la elección del proveedor a escoger y donde la elección por proveedor puede ser 0-4 y se puede repetir el numero por lo que se usa la función sample, se define el numero de hormigas que se van a utilizar y la longitud de esta hormiga tiene que es el total de los 29 proveedores.

```
H=matrix(data=0,nrow=hormigas,ncol=pm)
  for (prov in 1:pm)
  {
    H[,prov]=sample(x=ncomp,size=hormigas,replace = TRUE,prob=mT[,prov])
  }
```

En seguida se genera la matriz de costos, que como lo vimos en la Tabla 2, los parámetros que se tienen son g, p y c, donde el costo de cada proveedor es diferente en cada sistema que se tiene. En esta parte se muestra una matriz que es la suma de el total de componentes generados aleatoriamente y haciendo una suma, para mostrar el costo que tiene cada hormiga.

```
###Matriz de costo
  MatrizdePesos=numeric ()
   for (mp in 1:hormigas) {
            Hor=numeric()
            PesosHormiga=0

            for (hg in 1:pm) {
                  Hor=pesos[H[mp,hg]+1,hg]
                  PesosHormiga=Hor+PesosHormiga
            }
            MatrizdePesos=c(MatrizdePesos,PesosHormiga)
  }
  MatrizdePesos=matrix(data=MatrizdePesos,nrow=hormigas,ncol=1)
```

La siguiente es la matriz de confiabilidades que es donde se unen el ACO con el evaluador que es la Función Generadora Universal, que se le llama en el código Atotal, donde se manda llamar como una función lo que realiza el evaluador donde se coloca la disponibilidad que marca el problema como A*<0.95.

```
   A=numeric()
  for (xx in 1:hormigas) {
            HD=Atotal(H[xx,], comp,archivo1,qq)
            A=rbind(A,HD)
            print(xx)
           }
CosyDisp=MatrizdePesos+ifelse(A<0.95,(0.95-A)*1000,0)
  MatrizdePesos=CosyDisp
```

Primero se genera una matriz de ceros con el total de hormigas y de proveedores, con la finalidad que esta matriz sea la matriz que haga la suma de todas las hormigas con respecto a su proveedor elegido. Esta elección se hace con el recorrido de cada hormiga, y con respecto a la matriz de costo que tenemos de cada hormiga, que se le llamo MatrizdePesos, se realiza la operación de 1/MatrizdePesos donde el resultado se coloca en la posición donde la hormiga eligió ese proveedor, esto se hace con todas las hormigas dependiendo de sus elecciones se van sumando todas las elecciones hasta acabar con todas las hormigas de esa generación.

Al terminar esta sección de suma de las hormigas y sus elecciones, tenemos una matriz de probabilidades.

```
totalporhormigasTXY=matrix(0,length(ncomp),pm)
```

```
for (a in 1:hormigas)
 {
            matrizindividual=matrix(0,length(ncomp),pm)
            for (mi in 1:pm)
                {
                    matrizindividual[H[a,mi]+1,mi]=1/MatrizdePesos[a,1]

                }

            totalporhormigasTXY=matrizindividual+totalporhormigasTXY
 }
```

Por último, se realiza la matriz de feromonas que se aplicará también la evaporación 1-p para multiplicarlo por la matriz de probabilidades de elegir cada lugar en los proveedores, que esto servirá que, si se encuentra esa posición ocupada, es porque es la elección posible que la siguiente generación podrá usar, después la evaporación se le suma al totalporhormigasTXY que esta suma pasará a ser la nueva probabilidad de elección para la siguiente generación.

```
 f=0.1
 p=0.01
 Txy=numeric()
 evap=1-p
evaporacion=evap*mT
 TXY=evaporacion+totalporhormigasTXY
return(list(TXY,H,MatrizdePesos))
}
```

Toda la generación del ACO se guardó como una función llamada HormigasTXY que retorna una lista, donde [[1]] es la nueva matriz de probabilidades, [[2]] es una matriz que representa los elementos elegidos, de igual manera regresa [[3]] que es la matriz de costos de esta generación, y por último [[4]] regresa el resultado de la evaluación por hormiga.

4.3 Código UGF

Para generar la función de evaluación que se hace con la función generadora universal que como se menciona en el ACO, se le llama Atotal, ya que es una función donde se encuentra el evaluador por hormiga.

El problema comienza con la tabla 2 que es donde se presentan todos los tipos de proveedores para cada componente, dentro de la lectura que se hace de esta información se tiene que completar P y G ya que los diferentes estados que puede tomar G es 0 y el que se muestra en la tabla, al igual que P se realiza la resta de uno menos lo que nos da como elementos de entrada de P, y esto se hace para todos los componentes de modo que cada proveedor tendrá como elementos P Y G con sus complementos.

```
completar.PyG=function(comp)
{
 Utodas=list()
 cc=length(comp)
 for (i in 1:cc) {
    P=cbind(1-comp[[i]][,2],comp[[i]][,2])
  G=cbind(0,comp[[i]][,1])
  Tl=cbind(G,P)
  TT=list(Tl)

  Utodas=append(Utodas,TT)
 }
 return(Utodas)
}
```

De modo que NV se le llama al vector de componentes que se van a elegir de todos los proveedores, como se tiene que la hormiga es un vector que no se define la separación que se le da, la siguiente función separa cada uno de los elementos dependiendo de que número es el que le corresponde con respecto a la tabla 2, que en el primer componente corresponde con 7 proveedores, el segundo corresponde con 5 proveedores y así sucesivamente fijándonos en la tabla 2.

```
NVconcentrado=function(Utodas,nuvec)
{
 NSub=length(Utodas)
 lon.nuvec=rep(0,NSub)
 for(k in 1:NSub)
 {
  lon.nuvec[k]=dim(Utodas[[k]])[1]
 }
 acum.lon.nuvec=cumsum(lon.nuvec)
 acum.lon.nuvec=c(0,acum.lon.nuvec)
```

```
 nvconcentrado=list()

 for (d in 1:NSub) {

  nv=nuvec[(acum.lon.nuvec[d]+1):acum.lon.nuvec[d+1]]
  nv1=list(nv)
  nvconcentrado=append(nvconcentrado,nv1)
 }
 return(nvconcentrado)
}
```

Dentro de “ufun” se encuentra la selección de las listas para P y G que se le llamo Lp y Lg, para dar razón que son las listas a las que pertenecen, dentro de esta función tenemos dentro “funsa” que es una función ya existente en Rstudio, pero es para convertir una matriz en lista por renglones, ya que en su forma original lo hace por columnas, pero se adapto a lo que se requería, donde al final el resultado que retorna “ufun” es el resultado en paralelo.

```
funsa=function (X, FUN, ...,simplify = TRUE, USE.NAMES = TRUE)

{
 FUN <- match.fun(FUN)
 answer <- apply(X = X, FUN = FUN, MARGIN=1, ...)
 if (USE.NAMES && is.character(X) && is.null(names(answer)))
  names(answer) <- X
 if (!isFALSE(simplify))
  simplify2array(answer,(simplify == "array"))
 else answer
}
ufun=function(Utodas,numvec){
 #Configurar listas nuevas con numvec
 Veces=length(Utodas)
 N=length(numvec)
 Lp=list()
 Lg=list()
 for (f in 1:Veces) {

  lg=list(Utodas[[f]][,1:2]) #por que solo son dos estados
  lp=list(Utodas[[f]][,3:dim(Utodas[[1]])[2]])

  Lp=append(Lp,lp)
  Lg=append(Lg,lg)

 }
 tmm=length(Lp)
 Resultado.paralelo=list()
 #da las salidas de u de cada componente como este su paralelo
 for (r in 1:tmm) {
```

```
  Lpext=Lp[[r]]
  Lpex=funsa(Lpext,list)
  Lgext=Lg[[r]]
  Lgex=funsa(Lgext,list)
  nmext=numvec[[r]]
  S=OPar(Lpex,Lgex,nmext)
  S1=list(S)
  Resultado.paralelo=append(Resultado.paralelo,S1)
 }
 return(Resultado.paralelo)
}
```

Como se menciona con anterioridad, para la realización en paralelo, se realizaron las siguientes funciones.

```
OPar=function(Lp,Lg,numvec)
{  #Configurar listas nuevas con numvec
 LP=list()
 LG=list()
 N=length(numvec)
 posicion=1

 for (K in 1:N) {
  contador=1
  while(contador<=numvec[K])
  {
   LP[[posicion]]=Lp[[K]]
   LG[[posicion]]=Lg[[K]]
   posicion=posicion+1
   contador=contador+1
  }
 }

 SV=sum(numvec)
 if(SV==0)
 {
        return(rbind(c(0,1),c(1,0)))
 }

 #Extraigo posiciones
 unew=rbind(LP[[1]],LG[[1]])

 if(SV == 1){
  return(unew)
 }else{

  for (i in 2:SV) {
   ap=unew[1,]
```

```
    ag=unew[2,]
    bp=LP[[i]]
    bg=LG[[i]]
    unew=Parall(ap,bp,ag,bg)
   }
   return(unew)
  }
}

Parall=function(ap,bp,ag,bg)
{
 g1=outer(ag,bg,"+")
 g=unique(as.vector(g1))

 tn=length(g)
 p=numeric()
 cm=as.vector(outer(ap,bp,"*"))
 for (i in 1:tn) {
  p[i]=sum(cm[g1==g[i]])
 }
 x=rbind(p,g)
return(x)
}

#Para hacerlos de manera individual
ParaleloP=function(a1,a2)
{
 p1=outer(a1,a2,"*")
 p11=as.vector(rbind(as.matrix(p1[,1]),0)+rbind(0,as.matrix(p1[,2])))

 return(p11)
}

ParaleloG=function(b1,b2)
{
 g1=outer(b1,b2,"+")
 g11=unique(as.vector(g1))

 return(g11)
}
Paralelo=function(ap,bp,ag,bg)
{
 P=ParaleloP(ap,bp)
 G=ParaleloG(ag,bg)
 a=rbind(P,G)
 return(a)
}
```

Posteriormente a tener los paralelos de los elementos que aplicara, por que puede ser el caso que se use un solo proveedor con un solo elemento, y ese no se le hace ninguna modificación, por eso se señala que es solo para los que aplique.

De modo que cada elemento del sistema queda como un solo elemento ya que el paralelo se quedo como un solo elemento de U, se hace el sistema recursivo en serie para los elementos del sistema con la función “USerie”, contando con la función UGPserie dentro de esta función para sacar primero de manera recursiva dos elementos, y después dentro de la función principal “USerie” ya se puede hacer esto tantas veces se necesite, que en el caso de este problema se hace cuatro veces el sistema recursivo, dando como resultado o como retorno de la función una sola U, o como se le nombra en el programa “UTotal” que es una lista que contiene las P y G de la U total resultante.

```
USerie=function(lu){
 S1=UGPserie(lu[[2]][2,],lu[[1]][2,],lu[[2]][1,],lu[[1]][1,])
 UTotal=S1
 tam=length(lu)
 if(tam == 1){
 return(S1)
 }else{
 for (k in 3:tam) {
   UTotal=UGPserie(lu[[k]][2,],UTotal[2,],lu[[k]][1,],UTotal[1,])
 }
 return(UTotal)#LPGS
 }
}
#####FUncion para sacar G y P de manera recursiva de dos elementos
UGPserie=function(a,b,c,d){
 ex=expand.grid(a,b)
 g1=apply(ex,1,min)
 g=unique(g1)

 tn=length(g)
 p=numeric()
 cm=as.vector(outer(d,c,"*"))
 for (i in 1:tn) {
   p[i]=sum(cm[g1==g[i]])
 }
 U=rbind(p,g)

return(U)
}
```

El siguiente paso se realiza las demandas y la disponibilidad que ya están definidas en el problema, se observan en la tabla 1, las demandas “w” y disponibilidad “q”.

```
Demandas.w=function(lu,w){
 vecs.w=length(w)
 vecs.u=length(lu)
 t1=list()
 mw=funsa(matrix(w),list)
 for (i in 1:vecs.w) {
  for (y in 1:vecs.u) {
   s=sum((lu[[y]][1,])[lu[[y]][2,]>=mw[[i]]])
   ls=list(s)
   t1=append(t1,ls)

  }
 }
 vector.w=unlist(t1)
 l.lu=length(lu)
 l.w=length(w)
 lon.lu=rep(0,l.lu)
 for(k in 1:l.lu)
 {
  lon.lu[k]=length(lu)
 }
 acum.lon.lu=cumsum(lon.lu)
 acum.lon.lu=c(0,acum.lon.lu)

 w.concentrado=list()
 for (d in 1:(l.w)) {

  nw=vector.w[(acum.lon.lu[d]+1):acum.lon.lu[d+1]]
  nw1=list(matrix(nw))
  w.concentrado=append(w.concentrado,nw1)
 }
 dem=numeric()
 AT=list()
 for (m in 1:l.w) {
  #A.w=w.concentrado[[m]][n]
  dem=cumprod(w.concentrado[[m]])
  l=length(dem)
  A=dem[l]
  l.A=list(A)
  AT=append(AT,l.A)
 }
 return(AT)
}

####Disponibilidad
Dispo.q=function(d,q){

 lq=length(q)
 AT=list()
```

```
 for (t in 1:lq) {
   A=q[t] * d[[t]][1]
   l.A=list(A)
   AT=append(AT,l.A)
  }
  AF=sum(unlist(AT))
  return(AF)
}
```

Todas las funciones antes descritas quedan contenidas en una función general que es la que se utiliza en el ACO para la comunicación del evaluador y el optimizador que si podemos visualizar en el ACO se nombra como “Atotal” que se muestra a continuación y que es la recopilación de las funciones antes ya explicadas, que da como retorno la disponibilidad de un solo sistema que es el valor de A.

```
Atotal=function(nuvec,comp,qq)
{
Utodas=completar.PyG(comp)
numvec=NVconcentrado(Utodas,nuvec)
#son las u ya con su paralelo , si aplicara
lu=ufun(Utodas,numvec)
#Es el recursivo de las u para sacar U.final=UT
UT=USerie(lu)

#extraer w para demandas
w=qq[1,]

#Demandas
d=Demandas.w(lu,w)
#Disponibilidad
q=qq[2,]
dis=Dispo.q(d,q)
############Sistema total es:
A=dis
return(A)
}
```

5. Discusión de resultados

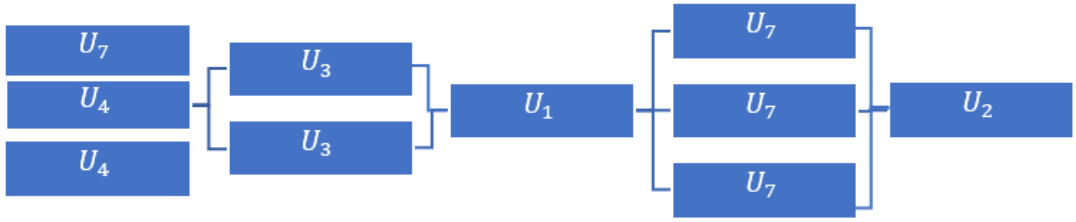
A partir del ejemplo que se menciona en el libro “The Universal Generating Function in Reliability Analysis and Optimization”, se menciona que los resultados obtenidos dentro de la tabla 3 hacen mención a la disponibilidad calculada , al igual que las estructuras de las posibles soluciones obtenidas mediante un costo mínimo por el Algoritmo genético, donde la estructura de cada componente se está ya representada por una cadena , donde el número de los elementos por proveedor se muestran en la tabla 3.

Tabla 3. Solución óptima para el problema de optimización de la estructura de MSS del ejemplo tomado del libro “The Universal Generating Function in Reliability Analysis and Optimization”.

	A*=0.95	**A*=0.97**	**A*=0.99**
Disponibilidad del sistema “A”	0.950	0.970	0.992
Costo del sistema “C”	9.805	10.581	15.870
Estructura del sistema			
Alimentadores primarios	1*7,2*4	2*2	2*4,1*6
Transportadores primarios	2*3	6*5	2*3
Apiladores-recuperadores	1*1	1*1	2*2,1*3
Alimentadores secundarios	3*7	6*9	3*7
Transportadores secundarios	1*2	3*3	3*4

Tomando en cuenta lo que se pedía en el libro que la disponibilidad debe de quedar 0.95 se tiene el siguiente diagrama de bloques que corresponde a la primera columna de la tabla 3, con un costo de 9.805.

ILUSTRACIÓN 3DIAGRAMA DE BLOQUES DEL EJEMPLO DEL LIBRO.



Donde en las ilustraciones 5 y 6 corresponden a las corridas para saber cuál era el parámetro por el que se iba a multiplicar A que es la disponibilidad, ya que si era mayor que 0.95, este se fuera a un número grande y así ya no fuera seleccionado la generación siguiente, por lo que se hicieron las corridas y cuando se multiplica por 1000 y por 100, no alcanza a llegar a un costo bajo, en comparación al del multiplicar A*10.

ILUSTRACIÓN 4 MULTIPLICACIÓN DE A *1000

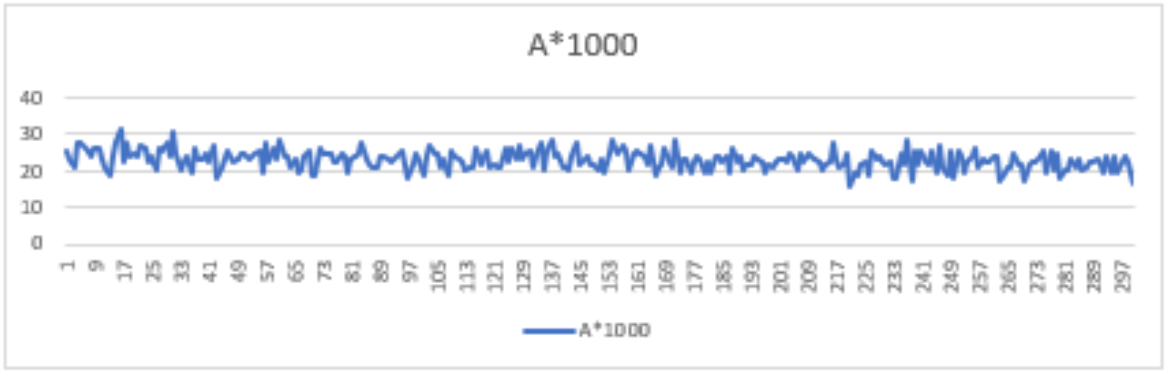
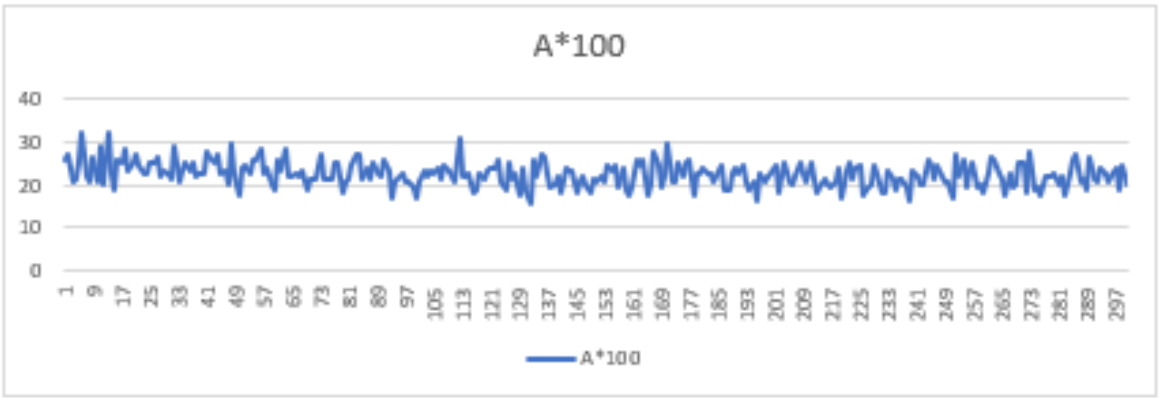
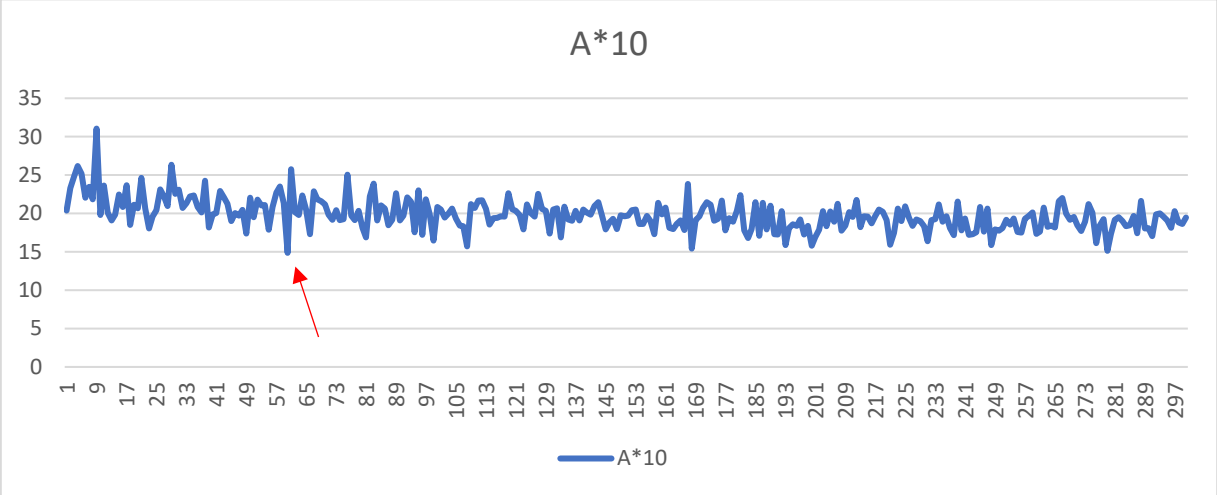


ILUSTRACIÓN 5 MULTIPLICACIÓN DE A*100



Solo para la elección de cual se encuentra el mejor costo, se hizo la corrida de 300 generaciones, donde podemos observar que se encuentra muy cerca del costo total de que habla en el libro, pero se continúan haciendo corridas con más generaciones. Por lo que. se hizo la elección de hacer la multiplicación de A*10 para ahí encontrar el costo mas bajo como se ve en la ilustración 6.

ILUSTRACIÓN 6 MULTIPLICACIÓN A*10



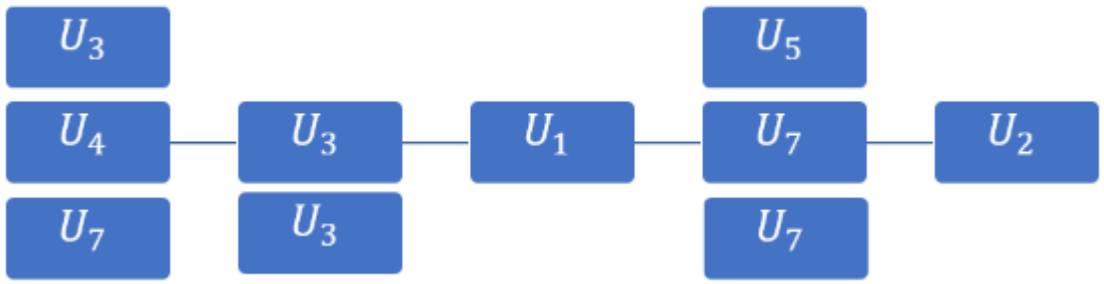
Donde el valor mínimo del costo es de 14.863 que en la ilustración 7 esta una pequeña marcación con una flecha de color rojo, en comparación de las ilustraciones 4 y 5 donde el valor mínimo del costo es de 15.532 y 15.775 respectivamente.

Se continuaron realizando más experimentos para encontrar la mejor solución posible, cambiando los parámetros del tamaño de hormigas y de generaciones, los experimentos que se realizaron fueron con 20,50 y 100 hormigas , donde se le daba la variante de generaciones de 100,200,500 y 1000 generaciones, donde el experimento de 100 hormigas y 500 generaciones llego al resultado de un costo $9.886 , con la siguiente elección de proveedores por componente; un componente del proveedor 3 y 4, y un componente del proveedor 7 para subsistema de alimentadores primarios, del proveedor 3 se requiere dos componentes para el subsistema de trasportadores primarios, donde al proveedor 1 se requiere un elemento para el subsistema de apiladores-recuperadores, él proveedor 5 se requiere un elemento y del proveedor 7 se requieren dos elementos para el subsistema de alimentadores secundarios, para el último subsistema que es de transportadores secundarios se requiere del proveedor 2 un solo elemento; donde el vector de entrada para el evaluador queda de la siguiente manera

(0,0,1,1,0,0,1,0,0,2,0,0,1,0,0,0,0,0,0,0,1,0,2,0,0,0,1,0,0), y el tiempo en el que se pudo llegar a este resultado fueron de 16.85 horas, teniendo en cuenta que esto para una confiabilidad $A \geq 0.95$.

Donde el siguiente diagrama de bloques muestra la combinación antes descrita.

ILUSTRACIÓN 7 DIAGRAMA DE BLOQUES DE MEJOR SOLUCIÓN PROPUESTA.



6. Conclusiones

La implementación del ACO, represento un reto, ya que tuvo que ser adaptado al problema de combinatoria de selección de proveedores con componentes repetidos. El ACO en combinatoria discreto es muy efectivo, pero carece de un mecanismo para el manejo de restricciones; así que se probaron varias propuestas. El uso de un factor penalización en el incumplimiento de la disponibilidad fue la que dio las mejores soluciones, como se vio en los resultados, siendo competitivo con la propuesta de Levitin y Lisnianski (2003) con el algoritmo genético; aunque no se pudo realizar una comparación en todos los aspectos, debido a que no proporciona la información suficiente, se trato en base a el problema propuesto , que el mismo ACO es un algoritmo con suficiente robustes para la cantidad de combinaciones que se deben de hacer y así mismo, tener en cuenta que buscar una solución factible, a partir del ACO para después que el evaluador nos indique si esa solución es la mejor y así seguir con el algoritmo, se habla que se realizan alrededor de $1.86x10^{20}$ combinaciones y si suponemos que una computadora pueda hacer al menos 1000 combinaciones por minuto le llevaría 67 501 años poder realizar las combinaciones posibles para encontrar el resultado esperado, de modo que con ayuda de este algoritmo, tratando de llegar a la solución óptima, hablando del tiempo que se pueda tardar en generar cada una de estas combinaciones, el tiempo pasa a no tomar importancia ya que lo que se quiere asegurar en este caso es que se llegue concretamente a que se respete el bajo costo y la disponibilidad deseada, que con lo antes explicado en esta tesis, pasamos a competir con los trabajos antes realizados de Levitin y Lisnianski (2003).

Dentro de la utilización de algoritmos los evolutivos, en este caso el de la colonia de hormigas y la unión de el evaluador que es la utilización de la Función Generadora Universal, todo esto para la minimización de costos en un diverso tipo de sistemas que se puedan representar como sistemas multi-estado, tratando siempre de ligarlos a una probabilidad de vida que sería la confiabilidad del sistema deseada, donde en este trabajo de tesis se cumple con el objetivo de la utilización de estas metodologías para los puntos requeridos que son costos y disponibilidad, que después se puede aplicar a diversos problemas dirigidos en esta misma línea.

7. Referencias

Alonso, S., Cordón, O., Fernández De Viana, I., & Herrera, F. (2004). La Metaheurística de Optimización Basada en Colonias de Hormigas: Modelos y Nuevos Enfoques. *Departamento de Ciencias de la Computación e Inteligencia Artificial, E.T.S. Ingeniería Informática, C/ Periodista Daniel Saucedo Aranda s/n, 1807*.

Arellano Martínez, Arnulfo, Huerta, M., & Rodrigo. (2021). Mejora de la confiabilidad en el edificio Valdés Vallejo de la UNAM. 52.100. http://132.248.52.100:8080/xmlui/handle/132.248.52.100/288

Grajales, D. H. M., Sánchez, Y. O., & Pinzón, M. A. A. (2006). La confiabilidad, la disponibilidad y la mantenibilidad, disciplinas modernas aplicadas al mantenimiento. Scientia et technica, 1(30), 155-160. https://doi.org/10.22517/23447214.6513

Anatoly Lisnianski, G. L., Lisnianski, A., & Levitin, G. (2003). Multi-state system reliability: assessment, optimization and applications. *Series on Quality, Reliability & Engineering Statistics*, *6*.

Goyal, N., Tyagi, S., & Ram, M. (2021). *Reliability Analysis of a System Using Universal Generating Function*. https://doi.org/10.1007/978-981-16-0037-1_7

Ushakov I (1998) *Optimal standby problem and a universal generating function*. Sov J Comput Syst Sci 25:61-73

Levitin, G., & Cai, K. Y. (2011). Guest editorial: Computational intelligence in reliability engineering. En *IEEE Transactions on Reliability* (Vol. 60, Número 2). https://doi.org/10.1109/TR.2011.2136570

The Universal Generating Function in Reliability Analysis and Optimization. (2005). En *The Universal Generating Function in Reliability Analysis and Optimization*. https://doi.org/10.1007/1-84628-245-4

The universal generating function in reliability analysis of binary systems. (2005). En *Springer Series in Reliability Engineering* (Vol. 3). https://doi.org/10.1007/1-84628-245-4_2

Gutjahr, W.J. (2000). A Graph-based Ant System and its convergence. Future Gener. Comput. Syst., 16, 873-888.