Centro de Investigación Científica y de Educación Superior de Ensenada, Baja California

Maestría en Ciencias en Ciencias de la Computación

Optimización multi-objetivo de un sistema de almacenamiento de datos en la multi-nube con un esquema de compartición de secretos basado en RRNS

Tesis
para cubrir parcialmente los requisitos necesarios para obtener el grado de Maestro en Ciencias

Presenta:

Luis Enrique García Hernández

Ensenada, Baja California, México
2019

Tesis defendida por

Luis Enrique García Hernández

y aprobada por el siguiente Comité

Dr. Andrey Chernykh

Miembros del comité

Dr. Edgar Leonel Chávez González

Dr. Israel Marck Martínez Pérez

Dr. Raúl Rivera Rodríguez

Dr. Ubaldo Ruiz López
Coordinador del Posgrado en Ciencias de la Computación

Dra. Rufina Hernández Martínez
Directora de Estudios de Posgrado

Resumen de la tesis que presenta **Luis Enrique García Hernández** como requisito parcial para la obtención del grado de Maestro en Ciencias en Ciencias de la Computación

Optimización multi-objetivo de un sistema de almacenamiento de datos en la multi-nube con un esquema de compartición de secretos basado en RRNS

Resumen aprobado por:

_____

Dr. Andrey Chernykh

Director de tesis

El almacenamiento como servicio (StaaS) es uno de los modelos de negocio del cómputo en la nube, en el cual una empresa alquila espacio en su infraestructura de almacenamiento a otra empresa o individuo. StaaS proporciona flexibilidad y escalabilidad para el almacenamiento de datos, sin embargo, trae varios problemas de ciberseguridad. En este trabajo, abordamos los métodos para mitigar los riesgos de confidencialidad, integridad, disponibilidad y fuga de información asociada con la pérdida de información y denegación de acceso. Nos basamos en un modelo adaptativo de almacenamiento de datos, soportado en esquemas de compartición de secretos y el sistema numérico de residuo redundante. Para adaptar los parámetros de configuración, tomamos en cuenta varios objetivos en conflicto: redundancia, probabilidad de pérdida de información y tiempo de extracción. Proponemos un enfoque basado en un algoritmo genético que es efectivo para resolver problemas de optimización multi-objetivo. Evaluamos los algoritmos NSGA-II, SPEA2 y MOCell, utilizando el marco de trabajo JMetal 5.6. Proporcionamos una evaluación experimental del enfoque propuesto utilizando datos reales de 11 proveedores de almacenamiento en la nube. Se determinó que el algoritmo MOCell es el que presentó los mejores resultados para nuestro problema, lo que nos permite obtener una mejor aproximación del Frente de Pareto. Concluimos que los algoritmos genéticos podrían utilizarse de manera eficiente para la toma de decisiones en sistemas de almacenamiento en un entorno de múltiples nubes.

Palabras clave: almacenamiento en la nube, optimización multi-objetivo, Sistema Numérico de Residuo, seguridad, esquemas de compartición de secretos.

Abstract of the thesis presented by **Luis Enrique García Hernández** as a partial requirement to obtain the Master of Science degree in Computer Science.

Multi-objective optimization of a multi-cloud data storage system with a secrets sharing scheme based on RRNS

Abstract approved by:

_____

Dr. Andrey Chernykh
Thesis Director

Storage as a service (StaaS) is one of the business models of Cloud Computing, in which a company rents space in its storage infrastructure to another company or individual. StaaS provides flexibility and scalability for data storage, however, it brings several cybersecurity issues. In this work, we address the methods to mitigate the risks of confidentiality, integrity, availability, information leakage associated with the loss of information, and denial of access. We rely on an adaptive secret sharing scheme and error correction codes based on the redundant residue number system. To adapt the con-figuration parameters, we take into account several conflicting objectives: redundancy, probability of information loss, and extraction time. We propose an approach based on a genetic algorithm that is effective for multi-objective optimization problems. We evaluated NSGA-II, SPEA2, and MOCell, using the JMetal 5.6 framework. We provide an experimental evaluation of the proposed approach using 11 real data cloud storage providers. It was determined that MOCell algorithm is the one that presented the best results for our problem, which allows us to obtain a better Pareto Optimal Front approximation. We concluded that genetic algorithms could be efficiently used for decision making in storage systems in a multi-cloud environment.

Keywords: Multi-objective optimization, cloud storage, Residue Number System, security, Secret Sharing Schemes.

Dedicatoria

A mi hijo que, con su mirada tierna y amorosa, hace que mi vida sea más bella.

A mi esposa, por su inmenso amor y su dedicación hacia nuestro amado hijo y a mí.

A mis padres por todo el apoyo y amor incondicional que siempre me han dado, por ser mis guías.

A mi hermana por su amor y ayuda constante.

A mi familia y amigos por su cariño y apoyo.

Al pueblo alegre y trabajador de este hermoso país que es México.

A Cuba, la tierra donde nací, que siempre la llevo dentro de mi corazón.

Agradecimientos

Al Centro de Investigación Científica y de Educación Superior de Ensenada (CICESE), en especial al Departamento de Ciencias de la Computación, que me acogió para realizar mis estudios de posgrado, por proporcionar las instalaciones y los medios necesarios para realizar mi investigación.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por brindarme el apoyo económico para realizar mis estudios de maestría. No. de becario 866993.

A mi director de tesis, Dr. Andrey Chernykh, por su gran ayuda, orientación y consejos durante todo este tiempo.

A los miembros de del comité de tesis, el Dr. Edgar Leonel Chávez González, el Dr. Israel Marck Martínez Pérez y el Dr. Raúl Rivera Rodríguez, por sus comentarios e indicaciones para mejorar mi investigación.

A mis compañeros y amigos de Ciencias de la Computación por su ayuda durante los cursos y la investigación.

A Rewer Miguel Canosa Reyes por recomendarme este gran centro que es el CICESE y por toda su ayuda.

A los profesores del Departamento de Ciencias de la Computación por sus excelentes clases y por el conocimiento transmitido.

Tabla de contenido

Lista de figuras

Lista de tablas

Capítulo 1. Introducción

La computación en la nube es un paradigma que permite ofrecer servicios de computación a través de una red, que usualmente es Internet. Su uso ha aumentado rápidamente en muchas organizaciones pues proporciona a los usuarios una larga lista de ventajas: capacidades de cómputo de provisión, acceso amplio y heterogéneo a la red; agrupación de recursos y rápida elasticidad con servicios medibles. Uno de los servicios destacados ofrecidos en la computación en la nube es el almacenamiento de datos en la nube, en el cual, los suscriptores en vez de almacenar los datos en sus propios servidores, los almacenan en los servidores del proveedor de servicios en la nube (CSP, *Cloud Service Provider*). Este servicio proporciona flexibilidad y escalabilidad para el almacenamiento de datos, sin tener que preocuparse por mecanismos de almacenamiento eficientes y problemas de mantenimiento, aunque surgen otros problemas de ciberseguridad.

Cuando los clientes dependen de un solo CSP se exponen a varios riesgos potenciales, como la falla en la disponibilidad del servicio, la integridad de datos y la privacidad. A pesar de que los CSP tienen regulaciones estándar e infraestructura poderosa para garantizar la privacidad de los datos del cliente y proporcionar una mejor disponibilidad, los informes de violación de la privacidad y la interrupción del servicio han sido evidentes en los últimos años. Con el fin solucionar este problema, Basescu et al. (2011) propusieron un modelo de nubes múltiples llamado internube, multi-nube o nube de nubes. Este modelo utiliza múltiples servicios comerciales de diferentes proveedores de manera transparente para los usuarios, es decir, se proporcionan como servicios de una nube única.

Existen numerosos estudios de seguridad de almacenamiento en la nube, como explican AlZain et al. (2011): “Garantizar la seguridad del cómputo en la nube es un factor de gran importancia porque los usuarios que almacenan regularmente información importante con sus respectivos proveedores, tienden a desconfiar de estos proveedores”. Esta desconfianza en los CSP es uno de los problemas de la computación en una sola nube que permitió que el entorno de múltiples nubes crezca en popularidad.

En este trabajo se presenta la optimización multi-objetivo de un modelo adaptativo de almacenamiento de datos en la multi-nube (Sección 2.3), basado en esquemas de compartición de secretos y el Sistema Numérico de Residuo Redundante. Dicho modelo permite incrementar la seguridad de la información, hace uso de distintos proveedores de almacenamiento en la nube para aumentar la

disponibilidad, confiabilidad y evitar el problema de bloqueo por el proveedor. En la **Figura 1** se muestra el flujo básico por el que pasa un secreto (archivo) para poder ser almacenado. Primero, el usuario elige el secreto que desea almacenar. Después, se le aplica algún algoritmo de compartición de secretos y se divide el secreto en n pedazos para ser almacenados en nubes distintas (las nubes utilizadas para almacenar los archivos que contienen los pedazos de información codificada se identifican en la **Figura 1** con una flecha más gruesa).

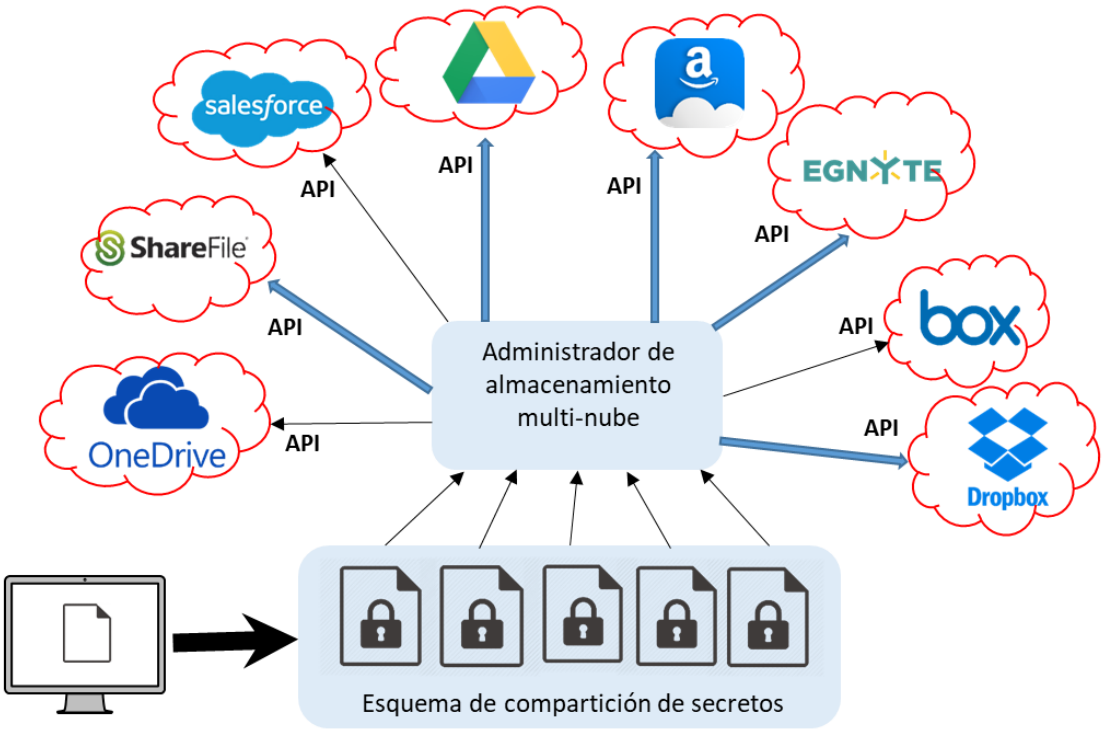

Figura 1. Carga de archivos codificados a la multi-nube.

Para poder recuperar el secreto (ver **Figura 2**), la cantidad de pedazos que se tienen que utilizar, depende de la configuración que se eligió en el esquema de compartición de secretos. En este trabajo, una configuración se identifica por la tupla (k, n) donde k representa la cantidad de pedazos requeridos para recuperar la información y n la cantidad de pedazos en los que se dividió la información. El ejemplo descrito anteriormente corresponde a una configuración $(3,5)$, en la cual guardamos la información en 5 nubes y luego solamente necesitamos que 3 de ellas estén

activas para recuperar la información original (las nubes desde las cuales se descargaron pedazos de información codificada se identifican en la **Figura 2** con una flecha más gruesa).

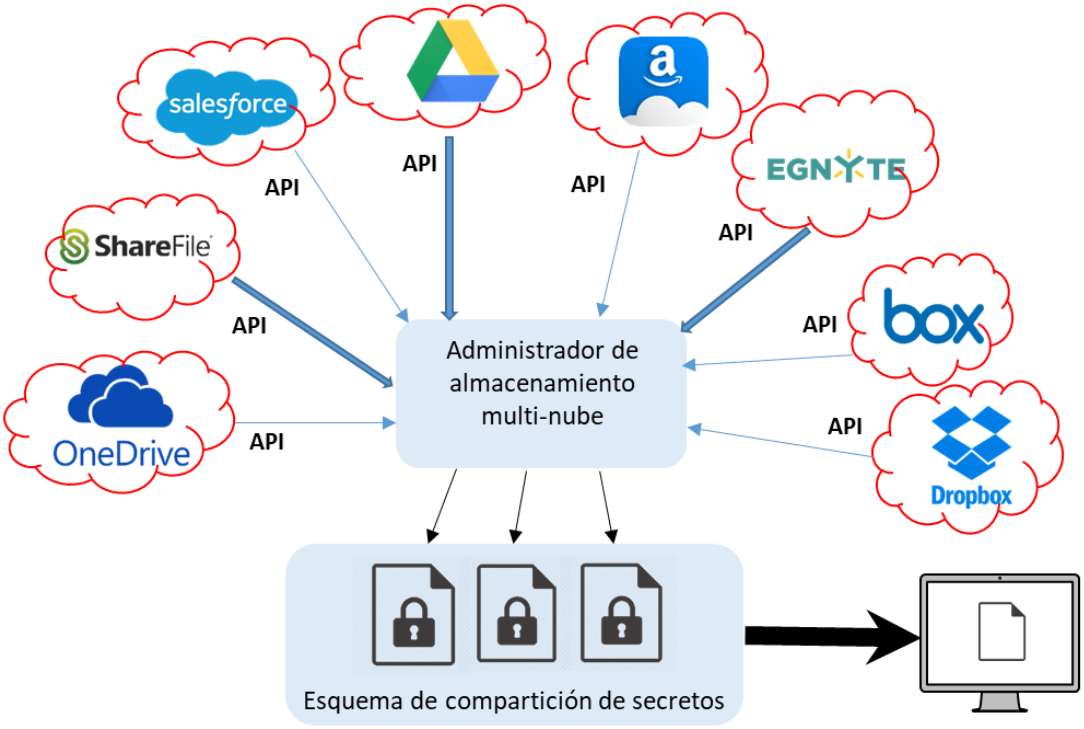

Figura 2. Descarga de archivos codificados de la multi-nube.

1.1 Antecedentes

En el modelo de almacenamiento en la nube, los datos se alojan en espacios de almacenamiento virtualizados, generalmente proporcionados por terceros accesibles a través de Internet. Hay muchos servicios de almacenamiento de datos que están disponibles de forma pública. Sin embargo, la confiabilidad, la seguridad y la calidad de servicio requeridas para el almacenamiento de datos a largo plazo siguen siendo problemas emergentes. En esta sección abordamos técnicas computacionales que han sido utilizadas para atacar dichos problemas de ciberseguridad y se incluye también una breve introducción a la optimización multi-objetivo aplicada a problemas de configuración.

1.1.1 Seguridad en el almacenamiento en la nube

Entre las técnicas más utilizadas para mejorar la seguridad en el almacenamiento en la nube están los esquemas de compartición de secretos (SSS, *Secret Sharing Scheme*), también conocidos como: "esquemas de umbral tipo (k, n)". Shamir (1979) presentó uno de los primeros esquema de umbral tipo (k, n), basado en la interpolación de polinomios. Poco tiempo después, de forma independiente, Blakley (1979) presentó un SSS basado en principios de la geometría finita. Luego en 1983, se publicaron dos esquemas de umbral tipo (k, n): Mignotte SSS (Mignotte, 1983) y Asmuth-Bloom SSS (Asmuth y Bloom, 1983), ambos utilizan aritmética modular y el teorema chino del residuo (CRT, *Chinese remainder theorem*). La ventaja del algoritmo de Asmuth-Bloom es que agrega ruido a la codificación para mejorar la seguridad, mientras que Mignotte SSS tiene entre sus ventajas que es computacionalmente estable y utiliza menos redundancia. La naturaleza distribuida de dichos esquemas ha posibilitado su utilización en el ambiente multi-nube.

Rabin (1989) propuso un algoritmo de dispersión de información (IDA, *Information Dispersal Algorithm*) que tiene similitudes con los sistemas de umbral tipo (k, n) y es eficiente en espacio, pero carece de confidencialidad ya que $k - 1$ pedazos pueden llegar a proporcionar información acerca de la información original.

También se han utilizado técnicas criptográficas para mitigar los problemas de confidencialidad y privacidad en el computo en la nube, como por ejemplo en el artículo de Marium et al. (2012), propusieron encriptar los datos con el algoritmo RSA y luego transferir con mayor seguridad combinando los protocolos de autenticación extendido y de autenticación por desafío mutuo.

Ermakova y Fabian (2013) presentaron una propuesta para el almacenamiento seguro de registros médicos en hospitales europeos, utilizando una arquitectura de múltiples nubes y SSS. En dicho trabajo se realiza una evaluación comparativa, teniendo en cuenta el rendimiento en tiempo, entre los algoritmos Shamir SSS y Rabin IDA.

DEPSKY (Bessani et al., 2013) es un prototipo de sistema que mejora la disponibilidad, integridad y confidencialidad de la información almacenada en la nube a través del cifrado, codificación y replicación de los datos en un ambiente multi-nube. Utiliza cuatro nubes comerciales, el algoritmo RSA con llaves de

1024 bits para encriptar las firmas de autenticación, SHA-1 para hash criptográfico, AES para encriptar la información, PVSS para distribuir la llave creada para la información a ser almacenada entre los distintos proveedores, y Reed-Solomon para verificación de errores.

Rathanam et al. (2014), propusieron un sistema de almacenamiento dinámico y seguro que utiliza la técnica de doble cifrado RSA y la técnica adaptativa Huffman para encriptar y comprimir la información antes de almacenarla. Su sistema permite realizar revisiones remotas a la información almacenada para comprobar su integridad.

Pundkar y Shekokar (2016), desarrollaron un sistema que utilizaba el algoritmo de Shamir SSS con el esquema (2,3) con la utilización de los servicios almacenamiento en la nube de Google Drive, Dropbox y OwnCloud para un portal estudiantil inter-campus de la Universidad de Mumbai. Dicho sistema permitía el almacenamiento de fotos y videos. Pienso que el punto débil de este sistema es que solamente utiliza tres proveedores de almacenamiento en la nube, lo cual le impide alcanzar mayores niveles de seguridad.

Babitha et al. (2016), presentaron una aplicación que cifra los datos utilizando la versión de 128 bit del estándar de cifrado AES y luego se cargan en una nube. El modelo propuesto utiliza un mecanismo de alerta basado en el servicio de mensajes cortos para informar los accesos no autorizados a los datos del usuario.

Celesti et al. (2016), plantearon un sistema que combina las propiedades del Sistema Numérico de Residuo Redundante (RRNS, *Redundant Residue Number System*) con el algoritmo de encriptación AES 256-bits. Cada pedazo de información generado al utilizar RRNS se encripta con AES 256-bits y luego es enviado a un proveedor de almacenamiento en la nube. De esta forma se ataca el problema de la colusión en la nube, se incrementa la seguridad de la información en términos de confiabilidad, integridad y confidencialidad. Entre las desventajas del sistema está el incremento de la cantidad de información a almacenar, así como también aumenta significativamente el tiempo de codificación / decodificación.

Chervyakov et al. (2017) concibieron un esquema de almacenamiento de datos llamado RRNS de rango aproximado (AR-RRNS), que combina las propiedades RRNS y SSS para dividir y distribuir secretos. Los autores utilizan estrategias de aproximación numérica para reducir el costo computacional del algoritmo. Mediante el análisis teórico, muestran que a través de la selección adecuada de los parámetros de RRNS, el sistema permite configurar la seguridad, la confiabilidad y la redundancia de datos.

Babenko et al. (2017) abordaron el enfoque de códigos de corrección de errores para mejorar el rendimiento de la infraestructura abierta de Berkeley para computación en red bajo la incertidumbre del comportamiento de los usuarios. Los autores utilizaron el conjunto de módulos de RRNS de la forma especial para corregir la injusticia del usuario y aumentar la confiabilidad de los datos, disminuyendo la redundancia y el tráfico de red.

Miranda-López et al. (2018) realizaron un análisis experimental del almacenamiento distribuido en la nube a través de 11 CSPs. Los autores utilizan los esquemas de umbral tipo (k, n) de Asmuth-Bloom y Mignotte. Evalúan las velocidades de carga-descarga y codificación-decodificación con diferentes configuraciones (k, n). La utilización de un sistema de almacenamiento distribuido basado en un esquema para compartir secretos y múltiples nubes, les permitió atacar los problemas de perdida de información, negación de acceso por un periodo de tiempo elevado, y fuga de información.

Tchernykh et al. (2018a) propusieron el algoritmo AC-RRNS, que es un esquema de compartición de secretos basado en RRNS que es computacional seguro y confiable. Resuelven el problema de la colusión en la nube mediante el uso simultáneo de las ideas detrás de Mignotte SSS y Asmuth-Bloom SSS asintóticamente ideal. En este trabajo se identificaron tres amenazas de seguridad principales: deliberada, accidental y colusión. Se enfocaron en el problema de colusión, cuando los adversarios pueden obtener acceso a partes de datos confidenciales en uno o varias nubes. Para reducir la incertidumbre de los riesgos de violaciones de seguridad de datos y la denegación de acceso a los datos, utilizaron códigos de localización y corrección de errores de RRNS.

El trabajo de Tchernykh et al. (2018b) presentó el algoritmo WA-RRNS que combina el esquema de acceso ponderado, SSS y RRNS con múltiples mecanismos de detección/recuperación de fallas, y de cifrado homomórfico. Para mejores compensaciones entre seguridad y rendimiento, WA-RRNS utiliza parámetros para ajustar la redundancia, la velocidad de cifrado-descifrado y la probabilidad de pérdida de datos. Su enfoque proporciona una forma segura de mitigar la incertidumbre del uso de almacenamientos en la nube. Sin embargo, la mejora alcanzada en la fiabilidad se proporciona a expensas del rendimiento reducido de la codificación / decodificación. WA-RRNS es tres veces más lento para las velocidades de codificación y cuatro veces más lento para las velocidades de decodificación en comparación con el sistema (k, n) tradicional, en promedio.

En la **Tabla 1** se presentan las principales características de los trabajos sobre seguridad en el almacenamiento en la nube mencionados en esta sección.

En este trabajo, proporcionamos mecanismos para la optimización multi-objetivo del proceso de configuración del modelo de almacenamiento de datos AR-RRNS, que fue propuesto por Chervyakov et al. (2017). Dicho modelo utiliza el esquema de umbral tipo (k, n) y RRNS, en el cual un archivo se divide en n pedazos de tal manera que con k pedazos o más, los datos se pueden recuperar. Para seleccionar los parámetros de configuración (k, n) del sistema y definir qué nubes específicas se utilizarán para el almacenamiento de datos, se tienen en cuenta tres objetivos (o criterios) de optimización: redundancia, probabilidad de pérdida de información y tiempo de extracción. Por lo tanto, podemos modelar el proceso de configuración de AR-RRNS como un problema de optimización multi-objetivo.

Tabla 1. Características de trabajos en la literatura.

Autor	**Multi-nube**	**SSS**	**Encriptación simétrica**	**Encriptación asimétrica**	**Análisis Teórico**	**Análisis Experimental**	**Seguridad**	**Interpolación**	**CRT**	**Almacenamiento de datos**	**Real CSPs**	**Privacidad**
Marium et al. (2012)				●		●				●		
Ermakova y Fabian (2013)	●	●				●				●		
Bessani et al. (2013)	●	●	●	●		●	●	●		●	●	●
Rathanam et al. (2014)				●		●				●		
Pundkar y Shekokar (2016)	●	●				●		●		●		
Babitha et al. (2016)			●			●	●			●		●
Celesti et al. (2016)	●	●	●			●	●		●	●	●	●
Chervyakov et al (2017)		●			●		●		●	●		●
Babenko et al. (2017)		●			●		●		●	●		●
Miranda-López et al. (2018)	●	●				●	●		●	●	●	●
Tchernykh et al. (2018a)		●			●		●		●	●		●
Tchernykh et al. (2018b)		●			●		●		●	●		●

1.1.2 Optimización multi-objetivo en problemas de configuración

La optimización multi-objetivo es un área de investigación que se ocupa de los problemas de optimización matemática que implican más de una función objetivo a optimizar simultáneamente. Se ha aplicado en muchos campos de la ciencia, incluida la ingeniería, la economía y la logística, donde se deben tomar decisiones óptimas en presencia de compensaciones entre dos o más objetivos en conflicto. A pesar de la gran variedad de técnicas desarrolladas en la Investigación de Operaciones y otras disciplinas para abordar estos problemas, la complejidad de su solución requiere enfoques alternativos.

Los algoritmos evolutivos (EA, *Evolutionary Algorithm*) han sido ampliamente utilizados para resolver problemas de optimización de múltiples objetivos, debido a que el enfoque poblacional de los EA permite encontrar múltiples soluciones óptimas de Pareto simultáneamente en una sola ejecución. En el libro de Coello et al. (2007) se brinda una visión general del campo de investigación denominado: optimización evolutiva multi-objetivo que se refiere al uso de algoritmos evolutivos de cualquier tipo (es decir, algoritmos genéticos, estrategias evolutivas, programación evolutiva o programación genética) para resolver problemas de optimización multi-objetivo (MOOP, *Multi-Objective Optimization Problem*). En dicho trabajo también se abordan otras metaheurísticas que se utilizan para resolver MOOP, por ejemplo: optimización de enjambre de partículas, sistemas inmunes artificiales, algoritmos culturales, evolución diferencial, colonia de hormigas, búsqueda tabú, búsqueda de dispersión y algoritmos meméticos.

Los algoritmos genéticos (GA) se encuentran entre las técnicas de inteligencia computacional más exitosas. Son metaheurísticas adecuadas y populares para resolver problemas de optimización multi-objetivo. La capacidad de los GA para buscar simultáneamente diferentes regiones de un espacio de solución hace posible encontrar un conjunto diverso de soluciones para problemas difíciles con espacios de soluciones no convexas, discontinuas y multimodales. Además, la mayoría de los GA multi-objetivo no requieren que el usuario priorice, incremente o asigne peso a los objetivos (Konak et al., 2006).

Goyal et al. (2012) abordaron la optimización multi-objetivo de un sistema de fabricación reconfigurable. Dicho sistema se basa en máquinas de herramientas reconfigurables, que tiene una estructura modular, consta de módulos básicos y auxiliares junto con el software de arquitectura abierta. Se analizaron dos objetivos a optimizar: el rendimiento y el costo, para la asignación óptima de la máquina para una línea de flujo de una sola pieza que permite el paralelismo de máquinas similares. La optimización

fue realizada en dos fases. En la primera, se aplica el algoritmo genético NSGA-II para obtener las soluciones no dominadas. En la etapa posterior, se emplea un enfoque de toma de decisiones de atributos múltiples para clasificar las soluciones del Frente de Pareto.

Guzek et al. (2014) estudiaron el problema de calendarización multi-objetivo de aplicaciones restringidas de precedencia en un sistema informático distribuido. Consideraron simultáneamente dos objetivos independientes: duración del cronograma y minimización del consumo de energía. Realizaron la adaptación de tres algoritmos genéticos (MOCell, NSGA-II e IBEA) mediante el diseño de nuevos operadores y su validación en términos de rendimiento y energía. MOCell fue el GA identificado como el más adecuado para este problema, a partir del análisis de tres indicadores de calidad: épsilon, distancia generacional invertida y propagación.

En el trabajo de Arora et al. (2015) se utilizó el algoritmo genético NSGA-II para realizar la optimización multi-objetivo de los parámetros de entrada de un generador termoeléctrico (TEG), que es esencial para diseñar dispositivos con criterios de rendimiento superiores. Los autores mostraron como optimizar simultáneamente tres objetivos en conflicto: potencia de salida, eficiencia térmica y función ecológica de TEG irreversible en los modos en serie y paralelo eléctricamente. En dicho trabajo se exploran los valores óptimos de parámetros de rendimiento / variables de diseño, contenidos en el Frente de Pareto mediante el uso de tres técnicas de toma de decisiones: Fuzzy Bellman-Zadeh, Shannon y TOPSIS.

Barry et al. (2015) analizaron el problema de optimizar el funcionamiento de las centrales hidroeléctricas utilizando modelos matemáticos, apoyados en conceptos informáticos en el diseño de los modelos y la configuración de estos modelos. Utilizaron un proceso de configuración basado en el algoritmo genético SPEA2 para explorar la relación entre la escala del modelo y su tiempo de resolución.

Zhu et al. (2015) abordaron el problema de calendarización del flujo de trabajo en entornos Infraestructura como Servicio (IaaS, *Infrastructure as a Service*), en el ámbito de la computación en la nube. El problema de optimización multi-objetivo planteado tiene dos criterios de optimización: tiempo total y costo. Para la resolución del MOOP utilizando algoritmos genéticos, realizaron la definición del cromosoma, el tipo de cruzamiento y de mutación. Evaluaron tres AGs: NSGA-II, MOEA/D and SPEA2 utilizando flujos de trabajo del mundo real y flujos de trabajo generados aleatoriamente. Los experimentos

realizados se basan en los tipos de instancia bajo demanda de Amazon EC2; sin embargo, el modelo propuesto es fácil de extender a los recursos y modelos de precios de otros servicios de IaaS.

Lu et al. (2016) presentaron un enfoque automático (denominado Zen-FIX) para recomendar, de manera óptima, soluciones para resolver no conformidades en línea de productos de software. Zen-FIX, fue implementado sobre la base de algoritmos genéticos, y para su evaluación se realizó un estudio con 52454 problemas de optimización de línea de productos de software, comparando siete algoritmos de búsqueda multi-objetivo. Los resultados muestran que el algoritmo genético MOCell superó al resto de los algoritmos evaluados: CellDE, IBEA, NSGA-II, PESA2, Random, SPEA2, en la mayoría de los problemas.

En esta tesis, proponemos un enfoque basado en algoritmos genéticos para la optimización multi-objetivo de la configuración del modelo de almacenamiento de datos en la multi-nube AR-RRNS.

1.2 Objetivos

1.2.1 Objetivo general

Desarrollar un algoritmo para optimización multi-objetivo de un sistema de almacenamiento seguro de datos en la multi-nube con un esquema de compartición de secretos basado en RRNS, considerando como objetivos de optimización: la probabilidad de pérdida de información, la redundancia y el tiempo de extracción.

1.2.2. Objetivos específicos

1. Desarrollar un algoritmo metaheurístico para solucionar el problema propuesto con enfoque de optimización multi-objetivo, definiendo la representación del individuo y la función de evaluación de la aptitud.

2. Determinar los parámetros y las restricciones adecuadas que permitan al algoritmo entregar una solución(es) al problema, que indique la configuración (k, n) eficiente y el subconjunto de nubes a utilizar, para minimizar la probabilidad de pérdida de información, la redundancia y el tiempo de extracción.

3. Evaluar, mediante indicadores de calidad, los resultados entregados por varios algoritmos metaheurísticos, para determinar cuál es el mejor para resolver nuestro problema de optimización multi-objetivo.

Capítulo 2. Modelo de almacenamiento de datos en la multi-nube

En este Capítulo se describen las características esenciales del modelo de computación en la nube, los problemas de ciberseguridad asociados a dicho modelo y las principales técnicas de seguridad informática para el almacenamiento en la nube. Describimos también, el modelo de almacenamiento de datos AR-RRNS (Chervyakov et al., 2017), y formulamos su configuración como un problema de optimización multi-objetivo.

2.1 Cómputo en la nube

La computación en la nube proporciona la próxima generación de sistemas informáticos distribuidos altamente escalables basados en Internet en los que los recursos computacionales se ofrecen "como un servicio". La definición más utilizada del modelo de computación en la nube es introducida por el Instituto Nacional de Estándares y Tecnología (NIST) como: "Un modelo que permite acceso ubicuo, conveniente y bajo demanda a una red compartida de recursos computacionales configurables (ej., redes, servidores, almacenamiento, aplicaciones y servicios) que pueden ser entregados y distribuidos con un manejo mínimo o interacción con el proveedor de servicios" (Mell y Grance, 2011).

El modelo en la nube ha motivado a la industria y la academia a adoptar la computación en la nube para alojar un amplio espectro de aplicaciones que van desde aplicaciones de computación intensiva hasta servicios livianos. El modelo también es adecuado para pequeñas y medianas empresas porque ayuda a adoptar las tecnologías de la información (TI) sin inversiones iniciales en infraestructura, licencias de software y otros requisitos relevantes. Además, los gobiernos se interesan más en las posibilidades de usar la computación en la nube para reducir los costos de TI y aumentar el alcance de sus servicios entregados.

Los proveedores de computación en la nube ofrecen sus "servicios" de acuerdo con diferentes modelos, de los cuales los tres modelos estándar definidos por el NIST son: Infraestructura como servicio (IaaS, *Infrastructure as a Service*), Plataforma como servicio (PaaS, *Platform as a Service*) y Software como servicio (SaaS, *Software as a Service*).

2.1.1 Sistemas de almacenamiento en la nube

En los últimos años se ha vuelto muy popular el modelo de negocio en la nube conocido como almacenamiento como servicio (STaaS, *Storage as a service*), en el cual una empresa alquila espacio en su infraestructura de almacenamiento a otra empresa o individuo (Wu et al., 2010). Las organizaciones usan STaaS para mitigar los riesgos en la recuperación ante desastres, proporcionar retención a largo plazo, así como también, mejorar la continuidad y disponibilidad del negocio. Los clientes firman un acuerdo de nivel de servicio (SLA, *Service Level Agreement*) con el proveedor de STaaS. A través del SLA se documenta qué servicios proporcionará el proveedor (por ejemplo, alquilar espacio de almacenamiento en función de un costo por gigabyte almacenado y un costo por transferencia de datos) y se definen los estándares de servicio que el proveedor está obligado a cumplir. En la actualidad existen un gran número de sistemas de almacenamiento en la nube, algunos de los cuales son listados por orden alfabético en la **Tabla 2**.

2.1.2 Multi-nube

Con el objetivo de resolver el problema de que los usuarios dependan solamente de un proveedor de servicios en la nube, investigadores en los laboratorios de IBM (Basescu et al., 2011) publicaron un modelo en el cual conectaron múltiples nubes a algo que llamaron internube o nube de nubes. En dicho modelo, cada proveedor de servicios mantiene un estado y posibles transiciones de estado que son activadas por las acciones de los clientes. Si un proveedor de servicios falla por problemas físicos, todas las acciones de dicho proveedor se deshabilitan por tiempo indeterminado. Este modelo y sus algoritmos publicados hicieron posible el surgimiento de la multi-nube, la cual puede ser definida como el uso de múltiples servicios computacionales de diferentes proveedores de manera transparente, es decir que la interacción simule el uso de una sola nube.

El modelo multi-nube también se refiere a la distribución de activos en la nube (por ejemplo, software) en varios entornos de cómputo en la nube. Se diferencia de la nube híbrida en que se refiere a múltiples servicios en la nube, en lugar de múltiples modos de implementación (público, privado, heredado). Además, en un entorno multi-nube, la sincronización entre diferentes proveedores no es

esencial para completar un proceso de cómputo, a diferencia de la computación paralela o la computación distribuida.

Tabla 2. Ejemplos de proveedores de almacenamiento en la nube.

Nombre	**URL completo**
Alibaba Cloud	https://www.alibabacloud.com/
Amazon Drive	https://www.amazon.com/clouddrive
Amazon S3	https://aws.amazon.com/es/s3/
Box	http://box.com/
certain safe	https://certainsafe.com/
Dropbox	http://dropbox.com/
Egnyte	https://egnyte.com/
Elephant drive	https://home.elephantdrive.com/
FlipDrive	https://flipdrive.com/
Google Drive	https://www.google.com/drive/
Hubspot	https://www.hubspot.com/
iCloud	https://www.icloud.com/
IDrive	https://www.idrive.com/
Jumpshare	https://jumpshare.com/
JungleDisk	https://www.jungledisk.com/
Justcloud	http://www.justcloud.com/
MediaFire	https://www.mediafire.com/
Mega	https://mega.nz/
Microsoft One Drive	https://onedrive.live.com/
pCloud	https://www.pcloud.com/
Rackspace	https://www.rackspace.com/cloud
Salesforce	https://www.salesforce.com/
Sharefile	https://www.sharefile.com/
SpiderOak	https://spideroak.com/one/
Storegate	https://www.storegate.com/gl/
SugarSync	https://www2.sugarsync.com/
Sync	https://www.sync.com/
Windows Azure	https://azure.microsoft.com/en-us/services/storage/
Yandex Disk	https://disk.yandex.com/

2.1.3 Problemas de seguridad en cómputo en la nube

Del conjunto de servicios que se ofrecen como parte del cómputo en la nube, en este trabajo, nos enfocaremos en el almacenamiento en la nube. Cuando se almacena información sensible en la nube, se desea que se mantenga segura. Las instalaciones de almacenamiento en la nube se encuentran vigiladas en todo momento y los proveedores de servicio afirman que incluso ellos no pueden saber dónde se almacena la información de algún usuario en específico. Sin embargo, estas declaraciones no son suficientes para garantizar la seguridad de la información y cesar las dudas de los usuarios (Lopez-Falcon et al., 2018).

Para garantizar la seguridad en el cómputo en la nube, la solución disponible más atractiva para investigadores ha sido el uso de técnicas de criptografía. Al usar algoritmos de encriptación en los archivos antes de que sean almacenados en la nube, agrega una nueva capa de seguridad, sin embargo, se crean otros problemas que analizaremos a lo largo de este documento.

Otro problema de seguridad que se debe analizar es el de restricción a un solo proveedor, esto ocurre cuando un consumidor es dependiente de algún producto o servicio de un proveedor sin poder utilizar el de otro proveedor y obtener un trato similar. También es importante señalar que, si un usuario depende completamente de un solo proveedor, su información se encuentra expuesta a los fallos de este.

2.2 Seguridad informática

La seguridad informática o ciberseguridad es la protección de los sistemas informáticos contra el robo o daño de su hardware, software o datos electrónicos, así como contra la interrupción o la mala dirección de los servicios que brindan.

Una de las actividades de la seguridad informática es la de prevención a la interrupción de servicio, que se puede definir como una reducción temporal del rendimiento del sistema, ya sea por fallos del sistema al ser reiniciado o por algún fallo que genere la perdida permanente de información. El enfoque

de seguridad de este trabajo es el de interrupción de servicio, de tal manera que un sistema con menos interrupciones de servicio es más seguro.

2.2.1 Técnicas criptográficas

La criptografía se ocupa de las técnicas de cifrado o codificado, destinadas a alterar las representaciones lingüísticas de ciertos mensajes con el fin de hacerlos ininteligibles a receptores no autorizados. En cómputo en la nube, la criptografía se utiliza para mantener la información distribuida de forma segura. Las técnicas de encriptación de información se pueden clasificar en dos subconjuntos: simétricas y asimétricas.

En un esquema simétrico o de llave privada, existen dos funciones prácticamente similares llamadas encriptación y decodificación, también una llave que se mantiene como secreto a los no participantes. La llave se utiliza para encriptar y decodificar el mensaje; si la llave se compromete, cualquier persona puede descodificar el mensaje. Debido a que no es fácil compartir la llave sin que esta se comprometa, esta técnica es más utilizada cuando no se requiere compartir dicha llave, esto es, cuando la persona que genera la información es la misma que la utiliza.

En un esquema asimétrico o de llave pública, al igual que en el esquema simétrico, existe un método de encriptación, decodificación y una llave privada, pero también existe una llave pública. La llave pública se utiliza para encriptar el mensaje y la llave privada para descodificarlo (Paar y Pelzl, 2010).

El algoritmo simétrico más popular es el Estándar Avanzado de Encriptación (AES), está basado en permutaciones lineales y transformaciones. Existen tres diferentes versiones del algoritmo, AES-128, AES-192 y AES-256 donde los números representan el tamaño de la llave en bits (Daemen y Rijmen, 2003).

Por otro lado, el algoritmo asimétrico más popular es el Rivest, Samir y Adleman (RSA), el cual debido a su costo computacional, es mejor utilizado para encriptar información de pocos bits (Rivest et al., 1978).

2.2.2 Códigos de borrado

En la teoría de la codificación, los códigos de borrado (EC, *Erasure Codes*) son una técnica de recuperación y detección de errores, cambia un mensaje de k símbolos por un mensaje de n dónde $n > k$. La creación de mensajes se puede generalizar a una interpolación polinomial. Primero, se define un grupo finito F de tamaño al menos n. La entidad que envía el mensaje, enumera los símbolos que contienen desde 0 hasta $k - 1$, luego se realiza una interpolación de $p(x)$ de grado $k - 1$. Después, se construye un mensaje con $p(k), \dots, p(n - 1)$. Dicho mensaje se puede recuperar utilizando la misma interpolación si los primeros k llegaron a su destino exitosamente (Lin et al., 2014).

2.2.3 Códigos Reed-Solomon

Los códigos Reed-Solomon pueden detectar y corregir múltiples errores de símbolos. Al agregar t símbolos de verificación a los datos originales, Reed-Solomon puede detectar cualquier combinación de hasta t símbolos erróneos, o corregir hasta $\lfloor t \,/\, 2 \rfloor$ símbolos. También son adecuados para corregir errores de ráfagas múltiples de bits, ya que una secuencia de $b + 1$ errores de bits consecutivos puede afectar cuanto más dos símbolos de tamaño b. La elección de t depende del diseñador del código. Las aplicaciones más comunes para estos códigos son CDs, DVDs, Blu-ray, WiMAX, código QR y sistemas de almacenamiento como RAID 6. También son utilizados en comunicación satelital (Reed y Solomon, 1960).

2.2.4 Replicación dinámica

La replicación de información consiste en colocar de forma estratégica copias de dicha información para incrementar su disponibilidad, confiabilidad y tolerancia a fallas. Entre las estrategias de replicación destaca la replicación dinámica, que consiste en responder a cambios de parámetros dentro del sistema eligiendo qué, cuándo y dónde se va a replicar (Tos et al., 2015).

2.2.5 Esquema de compartición de secretos

En un esquema de compartición de secretos (SSS, *Secret Sharing Scheme),* dada una llave x un administrador la divide en $y_1, \dots, y_n$ pedazos llamados sombras, de tal manera que la llave puede ser reconstruida utilizando cualquier k sombras, además no se puede obtener ningún fragmento de información utilizando s $(s < k)$ o menos sombras, después de realizar la división, las sombras son distribuidas a los distintos participantes. Dicho sistema se llama esquema de umbral tipo (k, n). El tamaño de las sombras crece exponencialmente con el número de participantes. La esencia de este esquema lo hace popular en sistemas distribuidos tales como el cómputo en la nube.

2.2.5.1 Esquema de compartición de secretos "Shamir"

El primer esquema de este tipo fue diseñado por Adi Shamir (Shamir, 1979) y se basa en la propiedad de los polinomios que dice que dados k puntos en un plano de dos dimensiones $(x_1, y_1), \dots, (x_k, y_k)$ con distintas x_i, existe un y solo un polinomio $q(x)$ de grado $k-1$ tal que $q(x_i) = y \ \forall i$. Por lo tanto, a la hora de dividir la información D en sombras, se elige un polinomio de grado $k-1$ al azar $q(x) = a_0 + a_1 x + \cdots + a_{k-1}x^{k-1}$ tal que $a_0 = D$ y se evalúan $D_1 = q(1), \dots, D_i = q(i), \dots, D_n = q(n)$. Dado cualquier subconjunto k de los D's evaluados, junto con sus índices, se pueden encontrar los coeficientes de $q(x)$ mediante interpolación de polinomios y después evaluar $D = q(0)$.

2.2.5.2 Esquema de compartición de secretos "Maurice Mignotte"

En el esquema propuesto por Maurice Mignotte, el secreto es un número entero, a diferencia del sistema de Shamir (1979) en el que el secreto es un polinomio. En este esquema, se requieren $d_1, \dots, d_n$ primos relativos por pares, el secreto es un entero $S, a \leq S \leq b$, donde a y b son enteros dados y $0 < a < b$. Para obtener el esquema (k, n) de tipo umbral se toman $d_1, \dots, d_n$ tales que el producto de k de los d_j es mayor a b y el producto de $k-1$ de los d_j es menor a a. La información se divide de la forma $x_j = S \ mod \ d_j$, $1 \leq j \leq n$ y se recupera con el teorema chino del residuo (Mignotte, 1983).

2.2.5.3 Esquema de compartición de secretos “Asmuth-Bloom”

Al igual que el esquema propuesto por Mignotte (1983), Asmuth-Bloom es un esquema de compartición de secretos basado en el teorema chino del residuo y se define a continuación. Sean k y n dos números enteros positivos tales que $0 < k + 1 \leq n$ y dada una secuencia de números primos relativos $m_0, m_1, \dots, m_n$ que cumplen con las siguientes propiedades:

$$m_0 < m_1 < \cdots < m_n, \tag{1}$$

$$\prod_{i=1}^{k+1} m_i > m_0 \prod_{i=0}^{k-1} m_{n-i}. \tag{2}$$

Dado un secreto s y un número generado al azar r de tal forma que $s' = s + rm_0 < \prod_{i=1}^{k+1} m_i$. Se comparte s al hacer $s_i = s' \; mod \; m_i, \forall i: 1 \leq i \leq n$. Para reconstruir el secreto cualquier subconjunto de A participantes tales que $|A| \geq k + 1$ es suficiente, primero se utiliza el teorema chino del residuo para obtener $x \equiv s_i \; mod \; m_i, \forall i \in A$ y después se reduce $x \; mod \; m_0$ para obtener el secreto original (Asmuth y Bloom, 1983).

2.2.6 Sistema numérico de residuo

El sistema numérico de residuo (RNS, *Residue Number System*), basado principalmente en el Teorema Chino del Residuo, es utilizado para representar un número entero muy grande mediante el uso de números enteros más pequeños, esto permite la reducción en complejidad de ciertas operaciones computacionales.

RNS es descrito por Ferruccio Barsi et al. (1973) de la siguiente forma: “Dado un conjunto de k números enteros positivos primos relativos por pares $\{m_1 \; m_2 \dots m_k\}$ llamados módulos, los enteros no negativos X en el intervalo $[0, M)$, dónde $M = m_1 m_2 \dots m_k$, son únicamente representados por n-tuplas $x_1 \; x_2 \dots x_k$ de sus residuos modulo m_i $(x_i = |X|_{m_i}, i = 1,2, \dots, k)$.” El entero X en el intervalo $[0, M)$ se

puede obtener a partir de sus dígitos $x_i's$ con la formula $\left(\sum_{i=1}^{k} x_i \frac{M}{m_i} B_i\right) mod\ M$ donde B_i es el número tal que $B_i \frac{M}{m_i} + b_i m_i = 1$.

2.2.7 Sistema numérico de residuo redundante

El sistema numérico de residuo redundante (RRNS, *Redundant Residue Number System*) utiliza un conjunto de $n = (k + r)$-tuplas para representar un entero del intervalo $[0, M)$. Utiliza $m_1\ m_2 \dots m_k\ m_{k+1} \dots m_n$ módulos y $x_1, x_2, \dots, x_k, x_{k+1}, \dots, x_n$ dígitos, los $m_{k+1} \dots m_n$ y $x_{k+1}, \dots, x_n$ son llamados módulos redundantes y dígitos redundantes, respectivamente.

RRNS es utilizado para el almacenamiento de datos. En este caso, la seguridad depende de los parámetros de k y n que se vayan a utilizar, la selección apropiada de estos parámetros provee el nivel de seguridad computacional necesario. Por otro lado, las propiedades del sistema permiten la corrección y detección de errores de fallas que pudieran ocurrir. La confiabilidad del sistema depende de $r = n - k$. Mientras mayor sea el valor de r, más confiable es el sistema, aunque con mayor redundancia de datos.

2.3 Modelo de almacenamiento de datos AR-RRNS.

El modelo de almacenamiento de datos en la nube AR-RRNS, propuesto por Chervyakov et al. (2017) será explicado a detalle en esta sección, pues la meta de esta tesis es realizar la optimización multi-objetivo del proceso de configuración de dicho modelo. AR-RRNS se basa en sistemas RRNS configurables y confiables en entornos multi-nube para garantizar la seguridad, la solidez y la confidencialidad. La operación que consume más recursos en la implementación de RNS es la operación de división mientras se encuentran residuos de RNS de rango dinámico. Para aumentar la eficiencia del procesamiento de datos y disminuir el consumo de energía durante la codificación y decodificación de datos, se utilizan módulos RNS de una forma especial $2^b \pm \alpha$, que permiten encontrar un residuo de división con complejidad lineal. También propone la aproximación del rango (AR) que permite sustituir operaciones de búsqueda de residuos al tomar bits más altos de un número, basado en la función introducida para calcular el rango aproximado

del número RNS. Sobre la base de las propiedades del valor aproximado y las propiedades aritméticas de RNS, se propuso el método AR-RRNS para la detección de errores, la corrección y el control de los resultados computacionales.

2.3.1 Definición formal del problema

Considere un conjunto de N nubes $C = \{c_1, c_2, \dots, c_N\}$. Cada nube $c_j = \{u_j, d_j, err_j\}$ está caracterizada por la velocidad de subida u_j, velocidad de descarga d_j, probabilidad de falla err_j , $\forall j = \{1, \dots, N\}$.

Un RRNS con una configuración (k, n), donde la información $D = \{1,2, \dots, n\}$ se divide en n pedazos para ser almacenados. Cada pedazo $i = \{s_i\}$ tiene un tamaño s_i $\forall i = \{1, \dots, n\}$.

Se utilizará la siguiente notación:

D	Tamaño original de la información,
D_E	Tamaño de la información encriptada,
s_i	Tamaño del $i - ésimo$ pedazo original,
s_{Ei}	Tamaño del $i - ésimo$ pedazo encriptado,
u_j	Velocidad de subida de la $j - ésima$ nube,
d_j	Velocidad de descarga de la $j - ésima$ nube,
T_D	Tiempo de decodificación total,
T_E	Tiempo de encriptación total,
T_{up}	Tiempo de subida de la información encriptada,
T_{dow}	Tiempo de descarga de la información encriptada,
T_{s}	$T_E + T_{up}$,
T_{ex}	$T_{dow} + T_D$,
R	Redundancia,
p_i	$i - ésimo$ módulo,
$P_r(k, n)$	Probabilidad de pérdida de información en (k, n) RRNS,
err_j	Probabilidad de falla en la $j - ésima$ nube.

El problema que se aborda en el presente trabajo, para aplicar AR-RRNS, es: encontrar la configuración (k, n) y un subconjunto C' de nubes minimizando los tres objetivos siguientes: la probabilidad de pérdida de información ($P_r(k, n)$), la redundancia (R) y el tiempo de extracción (T_{ex}). El problema puede ser formalizado de la siguiente forma:

- Minimizar $\mathrm{P_r}(k, n)$, que se calcula como:

$$\mathrm{P_r}(k, n) = \sum_{A \in F_{n-k+1}}^{n} \prod_{j \in A} err_j \prod_{j^c \in A^c} (1 - err_{j^c}), \tag{3}$$

  donde el conjunto F_{n-k+1} es el conjunto de todos los posibles $n - k + 1$ subconjuntos de C, y A^c es el complemento de los subconjuntos A y C. La información se puede perder solo si $n - k + 1$ partes tienen error o se perdieron.

- Minimizar R, que es la razón del tamaño de la información codificada almacenada D_E y el tamaño original de la información D:

$$R = \frac{D_E}{D}. \tag{4}$$

- Minimizar T_{ex}, que representa que tan rápido descarga la información de cada una de las nubes y se decodifica para que pueda ser usada, se calcula como la suma del tiempo de decodificación T_D y el tiempo de descarga T_{dow} :

$$T_{ex} = T_D + T_{dow},\ T_{dow} = \sum_{i=n-k+1}^{n} \frac{\mathrm{S_{Ei}}}{d_i}. \tag{5}$$

Suponga que cada uno de los pedazos es descargado de forma secuencial de las nubes. El proceso de descarga termina cuando k pedazos son descargados de forma correcta.

Otras restricciones a las cuales está sujeto nuestro problema de optimización multi-objetivo AR-RRNS, para garantizar un umbral de seguridad, son:

a) Se deben usar al menos dos nubes para almacenar fragmentos de información: $n \geq 2$;

b) Se necesitan al menos dos nubes para reconstruir la información original: $k \geq 2$;

c) El número de nubes donde se almacena la información debe ser mayor o igual al número de nubes necesarias para recuperar la información: $k \leq n$.

d) Se realizaron estudios experimentales con hasta 30 nubes, en este trabajo asumiremos dicho valor máximo para n, o sea, $n \leq 30$. Aunque el modelo de almacenamiento AR-RRNS no establece un valor máximo para el parámetro de configuración n.

2.3.2 Análisis de los objetivos de optimización

La probabilidad de pérdida de información $P_r(k,n)$ está en conflicto con los otros dos criterios: redundancia (R) y tiempo de extracción (T_{ex}). Los tres objetivos deben ser minimizados en el modelo de almacenamiento de datos AR-RRNS. Para $P_r(k,n)$, ver **Figura 3**, las configuraciones que generan las peores soluciones (o sea, que es mayor el valor de la probabilidad de pérdida de información) es cuando los parámetros k y n se acercan (por ejemplo: $(4,4)$; $(9,10)$; $(10,11)$). Mientras que para R (ver **Figura 4**) y T_{ex} (ver **Figura 5**), estas configuraciones generan las mejores soluciones. En tanto la configuración que proporciona el mejor resultado para $P_r(k,n)$, es cuando los parámetros k y n se alejan, es la peor para los criterios R y T_{ex} (por ejemplo, $(2,10)$ o $(2,11)$).

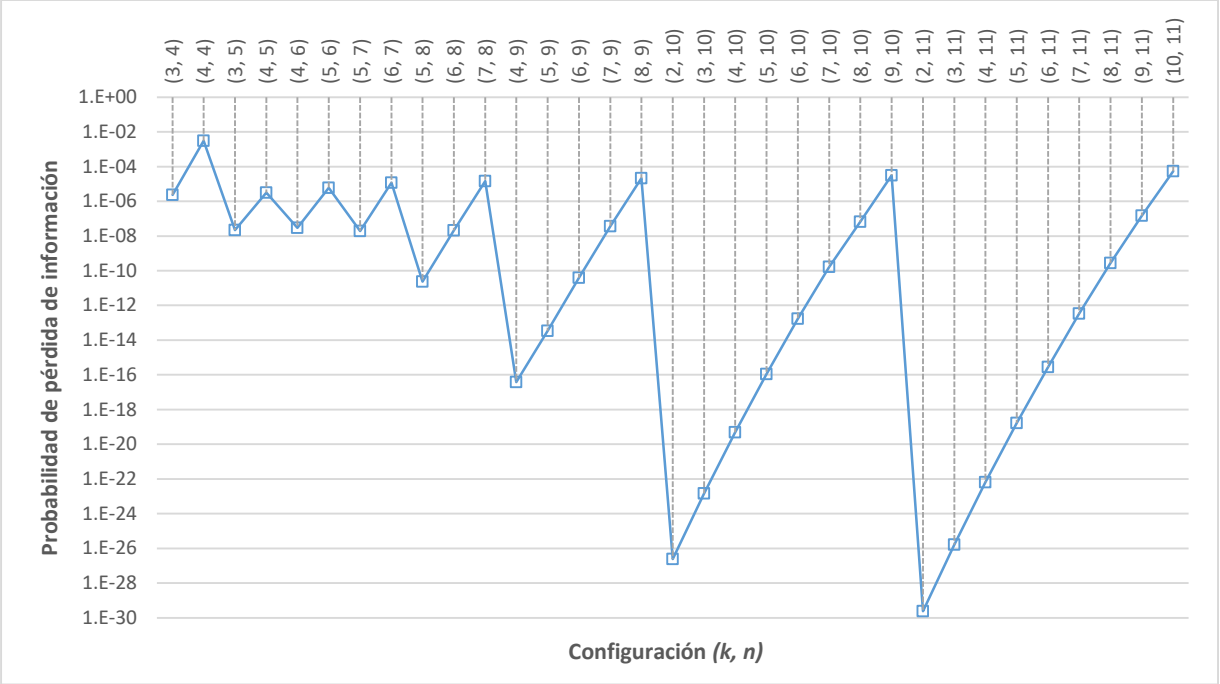

Figura 3. Probabilidad de pérdida de información para diferentes configuraciones (k, n) en AR-RRNS.

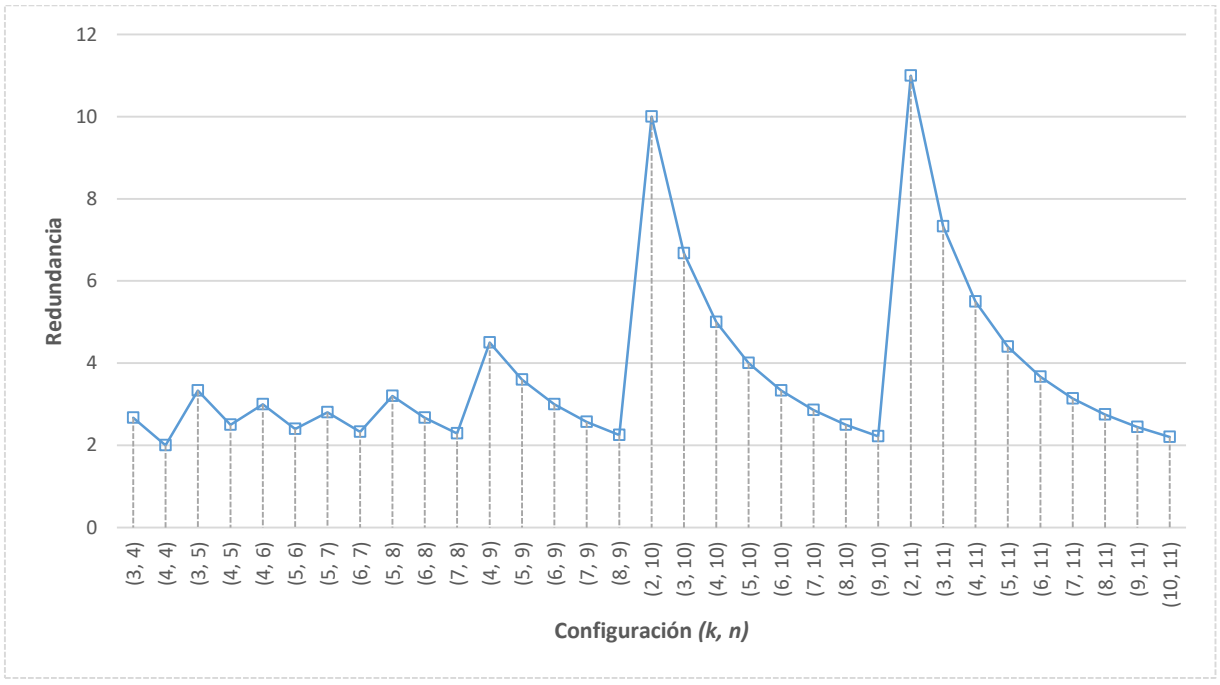

Figura 4. Redundancia para diferentes configuraciones (k, n) en AR-RRNS.

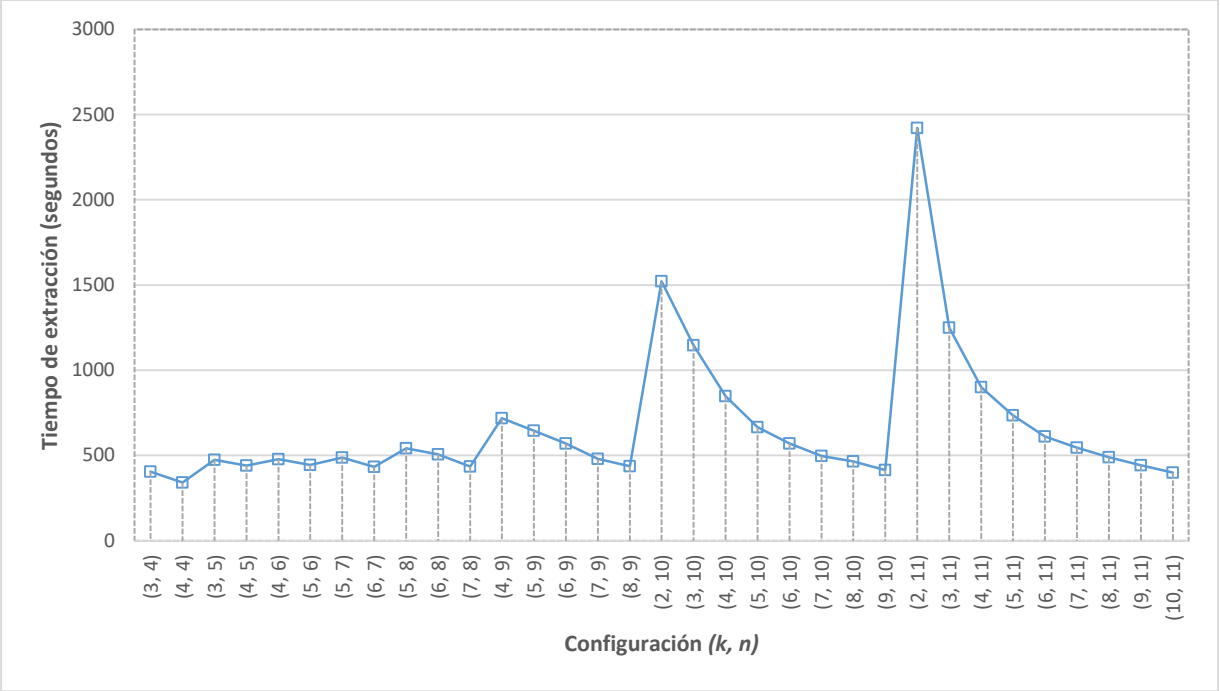

Figura 5. Tiempo de extracción para diferentes configuraciones (k, n) en AR-RRNS.

2.3.3 Proceso de almacenamiento y extracción de un archivo con AR-RRNS

En esta sección se describe el proceso de almacenamiento y de extracción de un archivo utilizando el esquema de compartición de secretos AR-RRNS.

El primer paso es encontrar, resolviendo el problema de optimización multi-objetivo descrito en la Sección 2.3.1, la configuración (k, n) y un subconjunto C' de nubes que minimicen tres criterios: la probabilidad de pérdida de información, la redundancia y el tiempo de extracción. Luego es necesario seleccionar el conjunto de $k + r$ números primos relativos por pares tales que $m_0 < m_1 < \cdots < m_k < m_{k+r}$. En RRNS con una configuración (k, n), $n = k + r$, donde r es la cantidad de módulos redundantes.

A continuación se calcula el rango dinámico del sistema, el cual es $M = m_1 \cdot m_2 \cdot \ldots \cdot m_n$. Se denomina $LenBlock(M)$ a la cantidad de bytes requeridos para representar M. Luego, ya que se tiene el archivo que se desea almacenar, se recorre el archivo de izquierda a derecha en intervalos de tamaño $LenBlock(M)$. Se generan los pedazos a almacenar de tal forma que el pedazo 1 contiene cada intervalo

$LenBlock(M)$ del archivo original mod m_1, el pedazo 2 cada intervalo mod m_2 y así sucesivamente. Luego se almacenan los pedazos generados en las n nubes seleccionadas en el primer paso. Cuando queremos recuperar el archivo original, debemos descargar k pedazos.

Se recorren los pedazos generados en intervalos de $LenBlock(M)\ mod\ m_i\ \forall\ i \in \{1, \dots, n\}$ para obtener los x_i, ya que se obtiene cada uno de los x_i se utiliza el teorema chino del residuo

$$\left(\sum_{i=1}^{k} x_i \frac{M}{m_i} B_i\right) mod\ M, \tag{6}$$

donde B_i es el número tal que $B_i \frac{M}{m_i} + b_i m_i = 1$. Para encontrar B_i se utiliza el algoritmo extendido de Euclides y dado que el conjunto de módulos m_i son primos relativos por pares, B_i existe y es único.

2.3.4 Ejemplo de codificación utilizando RRNS

Sea $\{11,13,17,19\}$ el conjunto de módulos que vamos a utilizar para ejemplificar un esquema $(k, n) = (2, 4)$; sean $m_3 = 17\ y\ m_4 = 19$ los módulos redundantes, esto quiere decir que vamos a utilizar el rango dinámico de los k módulos más pequeños $m_1 = 11\ y\ m_2 = 13$ que es $11 \cdot 13 = 143$.

Sea $X = 3014_{10} = 101111000110_2$ un secreto en su representación en base 10 y base 2 respectivamente. Debido a que solamente podemos representar números dentro del rango dinámico (143), y vamos a trabajar con representaciones en base 2, vamos a establecer el número base 2 previo al rango dinámico que es $2^7 = 128$, entonces vamos a dividir x de derecha a izquierda en segmentos de 7 bits, el segmento $S_1 = 1000110_2 = 70_{10}$ y el segmento $S_2 = 0010111_2 = 23_{10}$. Ahora aplicamos la operación de residuo para la cual utilizamos el símbolo mod, utilizando nuestro conjunto de módulos, sobre S_1 y S_2.

$$S_1 mod\ \{11,13,17,19\} = \{4,5,2,13\}_{10} = \{0100,0101,0010,1101\}_2,$$

$$S_2 mod\ \{11,13,17,19\} = \{1,10,6,4\}_{10} = \{0001,1010,0110,0100\}_2.$$

Para crear los pedazos que van a ser almacenados, vamos a concatenar los residuos base 2 de la siguiente manera

$$C_1 = S_{2,1}S_{1,1} = 00010100_2 = 20_{10}$$

$$C_2 = S_{2,2}S_{1,2} = 10100101_2 = 165_{10}$$

$$C_3 = S_{2,3}S_{1,3} = 01100010_2 = 194_{10}$$

$$C_4 = S_{2,4}S_{1,4} = 01001101_2 = 141_{10}$$

Si suponemos, por ejemplo, que los pedazos 2 y 3 fueron dañados, entonces para recuperar el secreto original, solamente tenemos los pedazos 1 y 4 (C_1, C_4), debido a que utilizamos el esquema $(2, 4)$ es posible recuperar el secreto original, si un pedazo más se hubiese dañado, no podríamos recuperar. Para recuperar, vamos a hacer uso del teorema chino del residuo que dice

$$X = \left(\sum_{i=1}^{k} x_i \frac{M}{m_i} B_i \right) mod\ M.$$

Para determinar el número B_1 y B_4, vamos a resolver las ecuaciones lineales:

$$\begin{aligned} \tfrac{M}{m_1} B_1 + b_1 m_1 &= 1 \\ \frac{209}{11} B_1 + b_1 11 &= 1 \\ 19(7) + (-12)(11) &= 1 \end{aligned} \tag{7}$$

y

$$\begin{aligned} \frac{M}{m_4} B_4 + b_4 m_4 &= 1 \\ \frac{209}{19} B_4 + b_4 19 &= 1 \\ 11(7) + (-4)19 &= 1\,. \end{aligned} \tag{8}$$

Como los módulos que estamos utilizando son primos relativos por pares, los números para resolver las ecuaciones existen y son únicos. Ya que solo contamos con los pedazos 1 y 4, el rango dinámico ahora es $M = m_1 * m_4 = 209$.

Para reconstruir el secreto original, debemos tener en cuenta que $C_1 = S_{2,1}S_{1,1}$ y $C_4 = S_{2,4}S_{1,4}$, vamos a sumar los residuos pertenecientes a S_1 con los de S_1 y los de S_2 con S_2, es decir $S_{1,1}$ con $S_{1,4}$ y $S_{2,1}$ con $S_{2,4}$ y tenemos las siguientes ecuaciones:

$$S_1 = S_{1,1}b_1M_1 + S_{1,4}b_4M_4$$

$$S_1 = (4 \cdot 7 \cdot 19 + 13 \cdot 7 \cdot 11)\ mod\ 209 \tag{9}$$

$$S_1 = 70_{10} = 1000110_2$$

y

$$S_2 = S_{2,1}b_1M_1 + S_{2,4}b_4M_4$$

$$S_2 = (1 \cdot 7 \cdot 19 + 4 \cdot 7 \cdot 11)\ mod\ 209 \tag{10}$$

$$S_2 = 23_{10} = 0010111_2\,.$$

Concatenamos S_2 con S_1 para obtener $S_2S_1 = 00101111000110_2 = 3014_{10}$ nuestro secreto original.

Capítulo 3. Optimización multi-objetivo

La optimización multi-objetivo es un área de toma de decisiones de criterios múltiples que se refiere a problemas de optimización matemática que involucran más de una función objetivo para ser optimizados simultáneamente. La optimización de múltiples objetivos se ha aplicado en muchos campos de la ciencia, incluida la ingeniería, la economía y la logística, donde se deben tomar decisiones óptimas en presencia de compensaciones entre dos o más objetivos en conflicto (Coello, 2017; Coello et al., 2007; Deb, 2014; Deb et al., 2004; Durillo et al., 2009; Frincu y Craciun, 2011). Un ejemplo sería: minimizar el costo al tiempo que maximiza la comodidad al comprar un automóvil.

Para un problema de optimización no trivial de objetivos múltiples, no existe una solución única que optimice simultáneamente cada objetivo. En ese caso, se dice que las funciones objetivas son conflictivas, y existe un número (posiblemente infinito) de soluciones óptimas de Pareto. Una solución se llama no dominada o Pareto óptima, si ninguna de las funciones objetivo puede mejorar su valor sin degradar algunos de los otros valores objetivos. Sin información de preferencia subjetiva adicional, todas las soluciones óptimas de Pareto se consideran igualmente buenas. Los investigadores estudian problemas de optimización multi-objetivo desde diferentes puntos de vista y, por lo tanto, existen diferentes filosofías y objetivos de solución al establecerlos y resolverlos. El objetivo puede ser encontrar un conjunto representativo de soluciones óptimas de Pareto, y / o cuantificar las compensaciones en el cumplimiento de los diferentes objetivos, y / o encontrar una solución única que satisfaga las preferencias subjetivas de la persona en el rol de tomador de decisiones.

3.1 Conceptos básicos de optimización multi-objetivo

Muchos problemas de búsqueda y optimización del mundo real se plantean naturalmente como problemas de programación no lineal que tienen múltiples objetivos en conflicto. Debido a la falta de técnicas de solución adecuadas, tales problemas se convirtieron artificialmente en un problema mono-objetivo y se resolvieron. La dificultad surgió porque tales problemas dan lugar a un conjunto de soluciones óptimas de compensación (conocidas como soluciones óptimas de Pareto), en lugar de una única solución óptima. Entonces se vuelve importante encontrar no solo una solución óptima de Pareto, sino la mayor

cantidad posible. Esto se debe a que cualquier par de estas soluciones constituye una compensación entre los objetivos, y los usuarios estarán en una mejor posición para tomar una decisión cuando se presenten dichas soluciones de compensación (Deb, 2014).

Un problema de optimización multi-objetivo (MOOP, *Multi-objective optimization problem*) se ocupa de más de una función objetivo. En la mayoría de los problemas prácticos de toma de decisiones, son evidentes objetivos múltiples o criterios múltiples. En el pasado, debido a la falta de metodologías de solución adecuadas, los MOOPs fueron proyectados y resueltos principalmente como problemas de optimización de un solo objetivo. Sin embargo, existen una serie de diferencias fundamentales entre los principios de funcionamiento de los algoritmos de optimización de un único objetivo y los algoritmos de optimización multi-objetivo, debido a que la solución de un MOOP debe intentarse utilizando una técnica de optimización de objetivos múltiples. En un problema de optimización de un solo objetivo, la tarea es encontrar una solución que optimiza la única función objetivo. Extendiendo la idea a la optimización de múltiples objetivos, se puede suponer erróneamente que la tarea en una optimización multi-objetivo es encontrar una solución óptima que corresponda a cada función objetivo. Ciertamente, la optimización de objetivos múltiples es mucho más que esta simple idea.

Para comprender la optimización multi-objetivo, debemos comenzar analizando la definición de un problema de optimización mono-objetivo, que puede ser resumida como: una minimización (o maximización) de una función $f(\mathbb{x})$ sujeto a $g_i(\mathbb{x}) \leq 0$, $i = \{1, \dots, m\}$, y $h_j(\mathbb{x}) = 0$, $j = \{1, \dots, p\}$ $\mathbb{x} \in \Omega$. Una solución que minimice (o maximice) la función escalar $f(\mathbb{x})$ donde $\mathbb{x}$ es un vector de variables de decisión dimensión N, $\mathbb{x} = (x_1, \dots, x_N)$, para algún universo Ω, donde $g_i(\mathbb{x}) \leq 0$ y $h_j(\mathbb{x}) = 0$ representan las restricciones que se deben satisfacer mientras se optimiza $f(\mathbb{x})$. Ω contiene todas las posibles soluciones que pueden ser utilizadas para satisfacer la función y sus restricciones.

Un método que permita encontrar el óptimo global (que puede no ser único) para un problema con una sola función objetivo, se conoce como optimización global, y se puede definir para un caso de minimización (sin perder generalidad) como: Dada una función $f: \Omega \subseteq \mathbb{R}^n \to \mathbb{R}$, $\Omega \neq \emptyset$, para un $\boldsymbol{x} \in \Omega$ el valor de $f^* \triangleq f(\mathbb{x}^*) > -\infty$ se conoce como óptimo global si y solo si $\forall \mathbb{x} \in \Omega$: $f(\mathbb{x}^*) \leq f(\mathbb{x})$. Donde $\mathbb{x}^*$ es la solución mínima global, f es la función objetivo y el conjunto Ω es la región factible de $\mathbb{x}$. Por esta razón, para un problema de optimización mono-objetivo se puede encontrar una única solución, a diferencia de los MOOP, en los que es posible encontrar un conjunto de soluciones en algunos casos infinito que, al ser

evaluados, producen unos vectores cuyos componentes representan los valores de Ω en el espacio asociado a las funciones objetivo.

En términos matemáticos, un MOOP puede ser formulado como:

Minimizar/maximizar

$$f_k(\mathbb{x}),\ \ k = 1, \dots, n\,, \tag{11}$$

sujeto a:

$$g_i(\mathbb{x}) \leq 0, i = 1, \dots, m\,, \tag{12}$$

$$h_j(\mathbb{x}) = 0, j = 1, \dots, p\,, \tag{13}$$

$$\mathbb{x} \in \Omega\,. \tag{14}$$

Donde una solución $\mathbb{x}$ es una variable de decisión vectorial de orden N, $\mathbb{x} = [x_1, \dots, x_N]^T$, en algún universo (espacio de decisión) Ω. Generalmente, para un MOOP, existen varias soluciones óptimas con diferentes niveles de compensación para las n funciones objetivo, que satisfacen las $m + p$ restricciones del problema y componen el conjunto óptimo de Pareto. En la mayoría de MOOP, se presentan limitaciones producidas por el manejo de los recursos disponibles para su ejecución o situaciones particulares en el ambiente o entorno de este, una solución al problema debe cumplir con cada una de estas limitantes, las cuales son conocidas como restricciones y pueden ser representadas matemáticamente como desigualdades, $g_i(\mathbb{x}) \leq 0$, o igualdades, $h_j(\mathbb{x}) = 0$. Por lo tanto, se cuenta con m restricciones de desigualdad y p restricciones de igualdad; si una solución $\mathbb{x}$ satisface todas las $m + p$ restricciones, es conocida como solución factible y el conjunto de soluciones factibles en Ω define una región factible conocida como espacio de búsqueda $\mathcal{S}$.

En los MOOP la meta es optimizar n funciones objetivo de forma simultánea. Los objetivos (o criterios) del problema deben ser minimizados o maximizados según lo defina cada problema, no obstante,

es común abordar problemas únicamente de minimización en cada uno de sus objetivos dada la posibilidad de modificar un problema con objetivos mixtos (minimización y maximización combinados) mediante el teorema de dualidad (Rao, 2009).

El vector de n funciones objetivo, $\mathbb{F}(\mathbb{x}) = [f_1(\mathbb{x}), \dots, f_n(\mathbb{x})]^T$, define un espacio coordenado (espacio objetivo) en el que cada solución puede ser representada; de forma que, se cuenta con un espacio de variables de decisión con dimensión N (espacio de decisión), en el que cada eje coordenado corresponde a los componentes del vector $\mathbb{x}$, y el espacio objetivo de dimensión n, en el que cada eje coordenado corresponde a un componente vectorial $f_k(\mathbb{x})$. Para cada solución $\mathbb{x}$ en el espacio de decisión existe un punto en el espacio objetivo el cual es obtenido a través de las funciones objetivo. La **Figura 6** ilustra estos dos espacios y una asignación entre ellos. Por lo tanto, se puede posible dividir el espacio de búsqueda en dos zonas no-superpuestas, una en la cual se encuentran las soluciones óptimas de Pareto (región óptima) y otra con las soluciones restantes. Generalmente se encuentran múltiples soluciones en la región óptima, por ello no se puede elegir una única solución para el MOOP cuando no se cuenta con información adicional sobre el problema. Por lo cual, es importante encontrar la mayor cantidad de soluciones Pareto óptimas como sea posible, lo que permite estructurar dos metas pertinentes en optimización multi-objetivo, primero encontrar el conjunto de soluciones más cercano a la región óptima real del problema, conocida como frente de Pareto y que a su vez sea lo más "diverso" posible tanto en el espacio de decisión como en el espacio objetivo, la diversidad está asociada a la distancia euclidiana entre las soluciones.

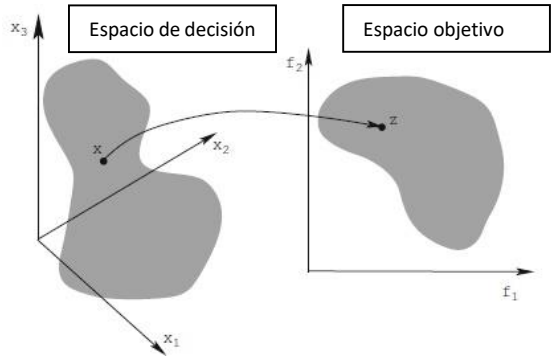

Figura 6. Representación del espacio variable de decisión y el espacio objetivo correspondiente.

3.2 Dos enfoques para la optimización multi-objetivo

Aunque la diferencia fundamental entre la optimización mono-objetivo y la multi-objetivo radica en la cardinalidad en el conjunto óptimo, desde un punto de vista práctico, el usuario necesita solo una solución, sin importar si el problema de optimización asociado es de objetivo único o de objetivos múltiples. En el caso de la optimización multi-objetivo, el usuario se encuentra ahora en un dilema. ¿Cuál de estas soluciones óptimas debe elegir? Para responder esta pregunta se puede poner el ejemplo del problema de compra de automóviles. Conociendo la cantidad de soluciones que existen en el mercado con diferentes compensaciones entre costo y comodidad, ¿qué automóvil se compra? Esta no es una pregunta fácil de responder. Involucra muchas otras consideraciones, como el financiamiento total disponible para comprar el automóvil, la distancia que se conducirá cada día, la cantidad de pasajeros que viajan en el automóvil, el consumo y el costo del combustible, el valor de depreciación, las condiciones de la carretera donde el automóvil se conducirá principalmente, la salud física de los pasajeros, el estatus social y muchos otros factores. A menudo, dicha información de nivel superior es no técnica, cualitativa y basada en la experiencia. Sin embargo, si un conjunto de soluciones de compensación ya está resuelto o disponible, uno puede evaluar los pros y los contras de cada una de estas soluciones en función de todas esas consideraciones no técnicas y cualitativas, pero aún importantes, y compararlas para hacer una elección. Por lo tanto, en una optimización multi-objetivo, idealmente el esfuerzo debe ser encontrar el conjunto de soluciones óptimas de compensación considerando todos los objetivos como importantes. Después de encontrar un conjunto de soluciones de compensación, un usuario puede usar consideraciones cualitativas de alto nivel para tomar una decisión. Por lo tanto, varios autores sugieren el siguiente principio para un procedimiento ideal de optimización multi-objetivo (Deb, 2014):

- Paso 1: Encuentre múltiples soluciones óptimas de compensación con una amplia gama de valores para objetivos.

- Paso 2: Elija una de las soluciones obtenidas utilizando información de nivel superior.

Cada solución de compensación corresponde a un orden específico de importancia de los objetivos. En el ejemplo de la compra de un automóvil, una de las soluciones es la que asigna más importancia al costo que a la comodidad, mientras que otra solución sería la que asigna más importancia a la comodidad que al costo. Por lo tanto, si se conoce un factor de preferencia relativa entre los objetivos

para un problema específico, no es necesario seguir el principio anterior para resolver un MOOP. Un método simple sería formar una función objetivo compuesta como la suma ponderada de los objetivos, donde la ponderación de un objetivo es proporcional al factor de preferencia asignado a ese objetivo en particular. Este método de escalarizar un vector objetivo en una única función objetivo compuesta convierte el MOOP en un problema de optimización de un solo objetivo. Cuando dicha función objetivo compuesta se optimiza, en la mayoría de los casos es posible obtener una solución de compensación particular. Este procedimiento de manejo de MOOP es mucho más simple, aunque sigue siendo más subjetivo que el procedimiento ideal anterior. Esta estrategia es conocida como optimización multi-objetivo basada en preferencias. Teniendo en cuenta la información de nivel superior, primero se elige un vector de preferencia. Posteriormente, el vector de preferencia se utiliza para construir la función compuesta, que luego se optimiza para encontrar una solución óptima de compensación mediante un algoritmo de optimización de objetivo único.

Los métodos clásicos de optimización multi-objetivo, que convierten múltiples objetivos en un solo objetivo mediante el uso de un vector de preferencia relativa de objetivos, funcionan de acuerdo con esta estrategia basada en preferencias. A menos que esté disponible un vector de preferencia confiable y preciso, la solución óptima obtenida por dichos métodos es altamente subjetiva para el usuario particular.

El procedimiento ideal de optimización multi-objetivo sugerido anteriormente es menos subjetivo. En el Paso 1, un usuario no necesita ninguna información relativa del vector de preferencia. La tarea allí es encontrar tantas soluciones de compensación diferentes como sea posible. Una vez que se encuentra un conjunto bien distribuido de soluciones de compensación, el Paso 2 requiere cierta información del problema para elegir una solución. En el enfoque ideal, la información del problema no se utiliza para buscar una nueva solución; en cambio, se usa para elegir una solución de un conjunto de soluciones de compensación ya obtenidas. Por lo tanto, existe una diferencia fundamental en el uso de la información del problema en ambos enfoques. En el enfoque basado en preferencias, se debe suministrar un vector de preferencia relativa sin ningún conocimiento de las posibles consecuencias. Sin embargo, en el enfoque ideal, la información del problema se utiliza para elegir una solución del conjunto de soluciones de compensación obtenidas.

3.3 Soluciones no dominadas y soluciones Pareto-óptimo

La mayoría de los algoritmos de optimización multi-objetivo utilizan el concepto de dominio en su búsqueda. En esta sección, definimos los conceptos de vector ideal, dominancia, optimalidad de Pareto y otros términos relacionados.

3.3.1 Vector objetivo ideal

Para cada una de las n funciones objetivo existe una solución óptima diferente. Un vector objetivo construido con estos valores objetivos óptimos individuales constituye el vector objetivo ideal:

$$\mathbb{x}^{*(i)} = \left[x_1^{*(i)}, x_2^{*(i)}, \dots, x_N^{*(i)}\right]^T. \tag{15}$$

De ahí que un $\mathbb{x}^{*(i)} \in \Omega$ es el que optimiza la k-ésima función objetivo $f_k(\mathbb{x})$, es decir:

$$f_k\left(\mathbb{x}^{*(i)}\right) = \underset{\mathbb{x} \in \Omega}{\acute{o}ptimo} \left(f_k(\mathbb{x})\right). \tag{16}$$

Por lo tanto el vector objetivo ideal $\mathbb{z}^*$ es:

$$\mathbb{z}^* = \mathbb{F}^* = [f_1^*, f_2^*, \dots, f_n^*]^T, \tag{17}$$

donde valor óptimo de la k-ésima función objetivo está representado por f_k^*, el punto en $\mathbb{R}^n$ que se obtiene a partir de $\mathbb{F}^*$ se considera la solución ideal para el problema propuesto.

El vector ideal es una solución que no existe, porque la solución óptima para un MOOP debe ser diferente para cada función objetivo y la única manera de que un vector ideal sea una solución factible, es cuando las soluciones óptimas son iguales; en este caso, las funciones objetivo no estarían en conflicto. A

partir de lo anterior, podemos definir un vector objetivo ficticio $\mathbb{z}^{**}$ cuyos componentes son estrictamente menores (para un caso de minimización) que el vector objetivo ideal, es decir $\mathbb{z}_k^{**} = \mathbb{z}_k^* - \beta_k$ con $\beta_k > 0$ para todo $i = 1,2, \dots, n$.

3.3.2 Conceptos de dominancia y Pareto

El concepto de dominancia ($\prec$) o dominancia de Pareto, permite comparar dos soluciones obtenidas (u y v) y concluir cuál de las dos domina al otro, lo que puede implicar cuál de los dos es mejor. Se define como: dados dos vectores $u = (u_1, \dots, u_n)$ y $v = (v_1, \dots, v_n)$, donde $u_k = f_k(\mathbb{x}')$ y $v_k = f_k(\mathbb{x})$, se dice que u domina v, expresado como $u \prec v$, si y solo si u es menor o igual a v (para un caso de minimización) pero u es estrictamente menor que v en al menos uno de sus componentes, i.e., $\forall k \in \{1, \dots, n\}, u_k \leq v_k \ \wedge \ \exists k \in \{1, \dots, n\}: u_k < v_k$. De lo anterior surgen tres posibilidades en la comparación de dos soluciones:

- u domina a v ($u \prec v$).
- u es dominado por v ($v \prec u$).
- u y v no son comparables, dado que uno no domina al otro y viceversa.

A partir de lo cual podemos señalar una propiedad de la dominancia: si una solución u no domina a una solución v, no implica que v domine a u. Deb et al. (2016) describen tres propiedades más del operador de dominancia:

- Reflexividad: la relación de dominancia es irreflexiva debido a que cualquier solución u no puede dominarse a sí misma.
- Simetría: la relación de dominancia es asimétrica debido a que si u domina a v no implica que v domine a u. Por lo que el caso contrario es cierto, si u domina a v entonces v no domina a u.

- Transitividad: la relación de dominancia es transitiva, lo cual significa que si u domina a v y v domina a w, entonces u domina a w.

Optimalidad de Pareto: Una solución $\mathbb{x} \in \Omega$ es óptima de Pareto con respecto a Ω si y solo si no existe un $\mathbb{x}' \in \Omega$ el cual $\mathbb{F}(\mathbb{x}') \prec \mathbb{F}(\mathbb{x})$, i.e. $\mathbb{F}(\mathbb{x}') = \big(f_1(\mathbb{x}'), \dots, f_n(\mathbb{x}')\big)$ domine $\mathbb{F}(\mathbb{x}) = \big(f_1(\mathbb{x}), \dots, f_n(\mathbb{x})\big)$. El término "óptimo de Pareto" hace referencia al espacio de decisión.

A partir de la optimalidad de Pareto, Coello et al. (2007) definen el conjunto de Pareto (conjunto de soluciones no dominadas $\mathcal{P}_S$ en el espacio de decisión) y el frente de Pareto ($\mathcal{P}_F$): el conjunto de soluciones no dominadas representadas a través de la evaluación de todas las posibles soluciones en Ω mediante las funciones objetivo plasmadas en el espacio objetivo.

Conjunto de Pareto: Para un MOOP dado, $f(\mathbb{x})$, el conjunto óptimo de Pareto, $\mathcal{P}_S$, se define como:

$$\mathcal{P}_S = \{\mathbb{x} \in S | \nexists \mathbb{x}' \in S \, f(\mathbb{x}') \prec f(\mathbb{x})\}. \tag{18}$$

Frente de Pareto: Para un MOOP dado, $f(\mathbb{x})$, y un conjunto óptimo de Pareto, $\mathcal{P}_S$, el frente de Pareto $\mathcal{P}_F$ se define como:

$$\mathcal{P}_F = \{f(\mathbb{x}) \in \mathbb{R}^n | \mathbb{x} \in \mathcal{P}_S\}. \tag{19}$$

Zitzler et al. (2002) establecieron dos definiciones que posibilitan comparar los conjuntos soluciones para un MOOP, para seleccionar la mejor aproximación al frente de Pareto:

- Dominancia fuerte ($\prec\prec$): una solución v es fuertemente dominada por una solución u ($u \prec\prec v$), si para las n funciones objetivo, u tiene un mejor rendimiento que v i.e. $\forall\, k \in \{1, \dots, n\}: u_k < v_k$.

- Dominancia débil ($\preccurlyeq$): un conjunto solución $\mathcal{P}_1$ es débilmente dominado por un conjunto $\mathcal{P}_2$ ($\mathcal{P}_2 \preccurlyeq \mathcal{P}_1$), si para las n funciones objetivo, ninguna de las soluciones en $\mathcal{P}_2$ domina fuertemente a algún miembro de $\mathcal{P}_1$.

El concepto de dominancia débil fue ampliado al establecer la relación “mejor que” ($\lhd$), tal que si $\mathcal{P}_2$ es al menos tan bueno como $\mathcal{P}_1$ pero $\mathcal{P}_1$ no es tan bueno como $\mathcal{P}_2$, i.e. $\mathcal{P}_2 \preccurlyeq \mathcal{P}_1 \ \wedge \mathcal{P}_1 \not\preccurlyeq \mathcal{P}_2$. Una dominancia débil indica que la cardinalidad del conjunto débilmente dominado ($|\mathcal{P}_1|$) es inferior o igual a la cardinalidad del conjunto ($|\mathcal{P}_2|$), debido a que debe contener al menos las mismas soluciones. Se define la dominancia débil ($\preccurlyeq$) partiendo de la relación $\lhd$ (mejor que) como: $\mathcal{P}_2 \preccurlyeq \mathcal{P}_1 \implies \mathcal{P}_2 \lhd \mathcal{P}_1 \ \vee \mathcal{P}_2 = \mathcal{P}_1$, por tanto, si $\mathcal{P}_2$ domina débilmente a $\mathcal{P}_1$, entonces $\mathcal{P}_2$ es mejor que $\mathcal{P}_1$ o son iguales.

3.4 Métodos de solución

Diferentes investigadores han definido el término “resolver un problema de optimización multi-objetivo” de varias maneras. Esta sección resume algunas y los contextos en los que se usan.

La resolución de un MOOP a veces se entiende como aproximación o cálculo de todo o un conjunto representativo de soluciones óptimas de Pareto. Cuando se enfatiza la toma de decisiones, el objetivo de resolver un MOOP se refiere a apoyar al tomador de decisiones en la búsqueda de la solución óptima de Pareto más preferida de acuerdo con sus preferencias subjetivas. La suposición subyacente es que se debe identificar una solución al problema para implementarla en la práctica. Aquí, un tomador de decisiones humano (DM, *Decision Maker*) juega un papel importante. Se espera que el DM sea un experto en el dominio del problema (Branke et al., 2008).

Los resultados más preferidos se pueden encontrar utilizando diferentes filosofías. Los métodos de optimización multi-objetivo se pueden dividir en cuatro clases. En los llamados métodos sin preferencia, no se espera que haya DM disponible, pero se identifica una solución neutral de compromiso sin información de preferencia. Las otras clases se denominan métodos a priori, a posteriori e interactivos y todos involucran información de preferencia del DM de diferentes maneras.

En los métodos a priori, primero se solicita información de preferencia al DM y luego se encuentra una solución que satisfaga mejor estas preferencias. En los métodos a posteriori, primero se encuentra un conjunto representativo de soluciones óptimas de Pareto y luego el DM debe elegir una de ellas. En los métodos interactivos, el tomador de decisiones puede buscar iterativamente la solución más preferida. En

cada iteración del método interactivo, el DM muestra las soluciones óptimas de Pareto y describe cómo podrían mejorarse las soluciones. La información proporcionada por el tomador de decisiones se tiene en cuenta al generar nuevas soluciones óptimas de Pareto para que el DM las estudie en la próxima iteración. De esta manera, el DM aprende sobre la viabilidad de sus deseos y puede concentrarse en soluciones que le interesan, además puede detener la búsqueda cuando lo desee (Marler y Arora, 2004).

3.4.1 Métodos sin preferencia

Cuando un tomador de decisiones no articula explícitamente ninguna información de preferencia, el método de optimización multi-objetivo puede clasificarse como método sin preferencia. Un ejemplo bien conocido es el método de criterio global, en el cual se busca encontrar una solución óptima de Pareto lo más cercana posible a $\mathbb{z}^*$. Se han planteado varias métricas para evaluar la "cercanía", una de las más utilizadas es la métrica L_p.

$$L_p(\mathbb{F}) = \left[\sum_{k=1}^{n} |f_k^* - f_k(\mathbb{x})|^p\right]^{1/p}. \tag{20}$$

En el método de criterio global, se trata de minimizar el valor de L_p para $1 \leq p \leq \infty$ tal que $\mathbb{x} \in \Omega$, con la meta de obtener una solución óptima de Pareto para cada valor de p. Dicho método es sensible a la escala de las funciones objetivo, por lo tanto, se recomienda que los objetivos se normalicen en una escala uniforme y sin dimensiones.

3.4.2 Métodos a priori

Los métodos a priori requieren que se exprese suficiente información de preferencia antes del proceso de solución. Los ejemplos bien conocidos de métodos a priori incluyen el método de función de utilidad, el método lexicográfico y la programación de objetivos. Por ejemplo, el método ordenamiento lexicográfico

utiliza la información del tomador de decisiones para definir una “clasificación ordenada” de las funciones objetivo. Realiza la búsqueda de una solución que sea la mejor posible (o el óptimo en algunos casos) para el primer objetivo de la “clasificación ordenada”, continuando con cada uno de los objetivos hasta el menos relevante. La solución al final de la ejecución, i.e. $\mathbb{x}_n^*$ para un problema de n objetivos, es presentada como la solución final al MOOP ($\mathbb{x}^*$) (Rao, 1984).

3.4.3 Métodos a posteriori

Los métodos a posteriori apuntan a producir todas (o un subconjunto representativo) las soluciones óptimas de Pareto. Los métodos a posteriori se dividen en dos clases: la primera son los métodos matemáticos basados en la programación a posteriori, donde se repite un algoritmo y cada ejecución del algoritmo produce una solución óptima de Pareto; y la segunda los algoritmos evolutivos (EA) donde una ejecución del algoritmo produce un conjunto de soluciones óptimas de Pareto.

Ejemplos de métodos a posteriori basados en programación matemática son: Intersección de límite normal (NBI) y métodos de Dominio de búsqueda dirigida (DSD), que resuelven el MOOP mediante la construcción de varias escalarizaciones. La solución a cada escalarización produce una solución óptima de Pareto, ya sea local o global, dichas escalarizaciones tienen el objetivo de obtener puntos de Pareto distribuidos uniformemente.

Los algoritmos evolutivos de optimización multi-objetivo son enfoques populares para generar soluciones óptimas de Pareto a un MOOP. Su principal ventaja es la capacidad de trabajar simultáneamente con un conjunto de posibles soluciones, conocido como población, los cuales pueden pertenecer al conjunto óptimo de Pareto y es posible alcanzarlos en una sola ejecución, a diferencia de un método tradicional donde en ejecuciones independientes se encuentran las soluciones. Además, los algoritmos evolutivos son menos susceptibles a la forma y continuidad del frente de Pareto.

3.4.4 Métodos interactivos

En los métodos interactivos para resolver MOOP, el proceso de solución es iterativo y el responsable de la toma de decisiones interactúa continuamente con el método cuando busca la solución más preferida. En otras palabras, se espera que el tomador de decisiones exprese preferencias en cada iteración para obtener las soluciones óptimas de Pareto que sean de interés para el tomador de decisiones y aprender qué tipo de soluciones son alcanzables.

Capítulo 4. Optimización evolutiva multi-objetivo

Los métodos clásicos de optimización multi-objetivo, por lo general, requieren aplicaciones repetitivas de un algoritmo para encontrar múltiples soluciones óptimas de Pareto y, en algunas ocasiones, tales aplicaciones ni siquiera garantizan el hallazgo de ninguna solución óptima de Pareto. Por el contrario, el enfoque poblacional de los algoritmos evolutivos (EA, *Evolutionary Algorithm*) es una forma eficiente de encontrar múltiples soluciones óptimas de Pareto simultáneamente en una sola ejecución de simulación. Este aspecto ha hecho que la investigación y las aplicaciones en la optimización evolutiva multi-objetivo (EMOO, *Evolutionary Multi-objective Optimization*) sean populares en las últimas décadas, como se puede constatar en el repositorio de 11,000 referencias a documentos científicos de EMOO (https://emoo.cs.cinvestav.mx/), que es archivado y mantenido por el Dr. Carlos A. Coello, del Centro de Investigación y de Estudios Avanzados (CINVESTAV, Instituto Politécnico Nacional, México) (Coello, 2017).

El cómputo evolutivo (EC, *Evolutionary Computation*) es un término genérico para varios métodos de búsqueda estocástica que simulan computacionalmente el proceso evolutivo natural. EC incorpora las técnicas de algoritmos genéticos (GA, G*enetic Algorithms*), estrategias de evolución (ES, *Evolution Strategies*) y programación evolutiva (EP, *Evolutionary Programming*), conocidas colectivamente como EAs. Estas técnicas se basan libremente en la evolución natural y el concepto darwiniano de “supervivencia del más apto”. Común entre ellos son la reproducción, la variación aleatoria, la competencia y la selección de individuos contendientes dentro de alguna población. En general, un EA consiste en una población de soluciones codificadas (individuos) manipuladas por un conjunto de operadores y evaluadas por alguna función de aptitud (Coello et al., 2007).

John Holland introdujo los algoritmos genéticos en 1960 basados en el concepto de la teoría de la evolución de Darwin; luego, su alumno David E. Goldberg extendió GA en 1989. Los algoritmos genéticos se usan comúnmente para generar soluciones de alta calidad para la optimización y los problemas de búsqueda, y son de especial interés para este trabajo de tesis; por lo tanto, profundizaremos más en estos que en otros EA. En la terminología de GA, una solución se llama individuo o cromosoma. Un cromosoma corresponde a una solución única en el espacio de la solución. El GA funciona con un conjunto de cromosomas llamado población. Forman la base de los sistemas evolutivos: 1) los operadores de variación (cruzamiento y mutación) que crean la diversidad necesaria dentro de la población; 2) la selección que

actúa como una fuerza que aumenta la calidad promedio de las soluciones en la población. Su combinación produce la mejora de la función objetivo.

Al ser un enfoque basado en la población, los GA son muy adecuados para resolver MOOP. La capacidad de GA para buscar simultáneamente en diferentes regiones de un espacio de soluciones permite encontrar un conjunto diverso de soluciones para problemas difíciles con espacios de soluciones no convexos, discontinuos y multimodales. El operador de cruzamiento de los GA aprovecha las estructuras de buenas soluciones con diferentes objetivos para crear nuevas soluciones no dominadas en partes inexploradas del frente de Pareto.

En este Capítulo presentaremos: los conceptos básicos de los algoritmos evolutivos, los principales algoritmos evolutivos multi-objetivo, los indicadores de calidad utilizados en EMOO para comparar los algoritmos y describiremos las principales características del framework jMetal para la investigación con algoritmos metaheurísticos.

4.1 Conceptos básicos de Algoritmos evolutivos (EA)

En esta sección se definen términos y conceptos estructurales básicos de EA; los “significados” de los términos descritos son normalmente análogos a sus contrapartes genéticas. Una estructura o individuo es una solución codificada para algún problema. Un individuo se representa como una cadena (o cadena de cadenas) correspondiente a un genotipo biológico. Este genotipo define un organismo individual cuando se expresa (decodifica) en un fenotipo. Un genotipo está compuesto por uno o más cromosomas, donde cada cromosoma está compuesto por genes separados que toman ciertos valores (alelos) de algún alfabeto genético. Un locus identifica la posición de un gen dentro del cromosoma. Por lo tanto, cada individuo decodifica en un conjunto de parámetros utilizados como entrada para la función en consideración. Finalmente, un conjunto dado de cromosomas se denomina población (Coello et al., 2007).

Al igual que en la naturaleza, los operadores evolutivos (EVOP, *Evolutionary Operators*) operan en la población de un EA, intentando generar soluciones con una aptitud cada vez mayor. Los tres EVOP principales asociados con EA son selección, cruzamiento y mutación. La **Figura 7** muestra un ejemplo de

un operador evolutivo de cruzamiento: “cruzamiento de un punto” (“*one-point crossover*” o “*single-point crossover*”), que opera en dos cadenas binarias principales; cada padre se corta y se cruza con una pieza del otro. Los individuos de la población por encima del promedio son seleccionados (reproducidos) para convertirse en miembros de la próxima generación con más frecuencia que los individuos por debajo del promedio. El EVOP selección les da a los individuos con mayor aptitud una mayor probabilidad de contribuir con uno o más descendientes en la siguiente generación.

Los cromosomas de valor real también se someten a estos mismos EVOP, aunque se implementan de manera diferente. Todos los EA usan algún subconjunto o variación de estos EVOP. Existen muchas variaciones en los operadores básicos; estos dependen de las restricciones del dominio del problema que afectan la estructura cromosómica y los alelos.

Un EA requiere tanto una función objetivo como una función de aptitud, que son fundamentalmente diferentes. La función objetivo define la condición de optimización del EA (y es una característica del dominio del problema) mientras que la función de aptitud (en el dominio del algoritmo) mide qué tan “bien” una solución particular satisface esa condición y asigna un valor real correspondiente a esa solución. Sin embargo, estas funciones son en principio idénticas (por ejemplo, en problemas de optimización numérica).

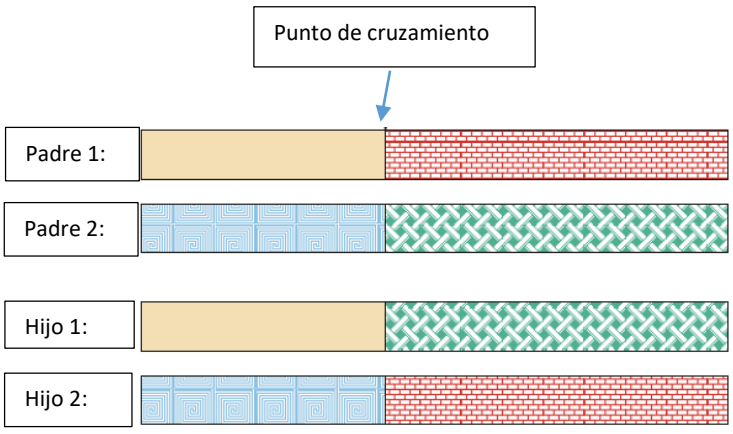

Figura 7. Ejemplo de un operador evolutivo: el “cruzamiento de un punto”.

Aunque existe mucho espacio para la creatividad al seleccionar y definir las instancias de EA (por ejemplo, representación genética y EVOP específicos), se debe considerar cuidadosamente el mapeo del problema al dominio del algoritmo. Las representaciones y/u operadores “inadecuados” pueden tener efectos perjudiciales sobre el rendimiento de EA.

La definición de EA que utilizaremos en ese trabajo es la siguiente: Dada una función objetivo (función de calidad o aptitud) que debe ser maximizada, se crea un conjunto de posibles soluciones (población de individuos) de manera aleatoria, como elementos del dominio de la función que al ser evaluados permite obtener de manera abstracta un valor de aptitud (el valor más alto es el mejor individuo). A partir de los valores de calidad, las mejores soluciones son seleccionadas como padres para la siguiente generación, para luego aplicarles algún operador genético de cruzamiento y/o mutación. El cruzamiento se aplica a dos o más candidatos seleccionados (padres), generando una o más posibles soluciones nuevas (hijos). La mutación tiene la capacidad de modificar cada posible solución generando nuevos candidatos (hijos modificados). Luego de aplicar cruzamiento y mutación se produce un conjunto de nuevas posibles soluciones (descendencia), que tienen un valor de aptitud que les permite competir por un lugar en la siguiente generación (supervivencia). Este proceso se repite hasta que un individuo alcance el valor de aptitud deseado (solución óptima) o llegue a tener suficiente calidad como para ser considerado como solución, también es posible definir algún limite computacional como: máximo número de iteraciones, cantidad de memoria o CPU utilizada, el valor máximo de aptitud se mantiene durante un largo periodo de tiempo (número de iteraciones o tiempo de ejecución) o cuando la diversidad de la población baja de un nivel definido.

Los procesos que componen un EA son, en su mayoría, estocásticos, por ejemplo, en la creación aleatoria de las posibles soluciones iniciales o en el proceso de selección, como los mejores individuos no se obtienen de manera determinística, lo cual facilita que los individuos “débiles” tengan alguna oportunidad de ser padres o sobrevivir para la siguiente generación. En el cruzamiento y la mutación, la magnitud y el lugar donde se realizará la modificación se define de manera aleatoria. Este tipo de decisiones permite aumentar la diversidad en la búsqueda, promoviendo el proceso de adaptación al entorno por parte de la población. El pseudocódigo genérico de un EA puede ser escrito así:

Algoritmo 1 Algoritmo evolutivo

1: **Inicialización**: Se crean n posibles soluciones (x) para una población inicial ($\mathbb{X}$). $x_1, \dots, x_n \in \mathbb{X}$
2: **Evaluación**: Se evalúan los candidatos con la función de aptitud. $x_i \rightarrow f_{aptitud}(x_i), i = 1, \dots, n$
3: **Mientras** (La condición de terminación ≠ verdadero)
4: **Seleccionar** los j padres de $\mathbb{X}$. $\mathbb{P} = \{p_1, \dots, p_j\}$
5: Crear los k hijos mediante **cruzamiento**. $\mathbb{H} = \{h_1, \dots, h_k\}$
6: **Mutar** h_i
7: **Evaluar** $h_i \rightarrow f_{aptitud}(h_i), i = 1, \dots, k$
8: Seleccionar individuos de $h_i \cup \mathbb{X}$ para la nueva generación $\mathbb{X}'$
9: $\mathbb{X} \rightarrow \mathbb{X}'$
10: **Fin**

4.1.1 Representación del individuo

Lo primero que debemos hacer durante el diseño de un EA es crear una estructura de datos que represente las características del problema y su contexto, diseñándola de forma tal que un sistema computacional pueda manipularla. O sea, toda la información inherente al problema y su contexto son los fenotipos y su codificación, crea una estructura de datos en un entorno de EA que contiene los genotipos. Existen varios EA, los cuales tienen diferencias como la representación de los individuos, específicamente para los GA es común utilizar caracteres de un alfabeto finito, en las ES se utilizan vectores de números reales, máquinas de estado finito para la programación evolutiva clásica y representación con árboles para la programación genética (Eiben y Smith, 2015).

4.1.2 Población

En EA una población es un multi-conjunto de cromosomas. La población es la unidad del proceso evolutivo, dado que los individuos no pueden evolucionar por sí solos, necesitan la interacción con sus pares para lograr cambios o adaptarse. Para iniciar una población solo es necesario definir cuántos individuos contiene, o sea, el tamaño.

Generalmente en las aplicaciones de EA el tamaño de la población permanece constante durante todo el proceso evolutivo, aumentando la competencia, dada la necesidad de sobrevivir en el recambio generacional. Con respecto a tipos de poblaciones, en implementaciones clásicas es común utilizar poblaciones de tipo panmíctica, en las que el apareamiento es libre y al azar (con igual probabilidad). Sin embargo, en algoritmos más sofisticados, típicamente se utilizan poblaciones distribuidas en islas o vecindarios, lo que reduce el número de individuos con los que se puede interactuar y se comparte información genética entre subpoblaciones cada cierto tiempo. El objetivo de esta limitación es aumentar la diversificación de la población en general e incrementar la explotación en los grupos de individuos que están distribuidos. La diversidad de la población representa la cantidad de individuos diferentes dentro de la población, y es un concepto muy importante a tener en cuenta en los EA.

4.1.3 Función de evaluación de aptitud

La función objetivo para un problema de optimización, que define las condiciones de optimalidad, puede ser la misma función de calidad (en el dominio del EA) o en algunos casos requerir alguna modificación, que permite evaluar el rendimiento de los individuos, es decir que tan “bien” el candidato satisface las condiciones, calculando un valor de aptitud para los genotipos que describe un mayor o menor nivel en los fenotipos, según sea el caso. La mayoría de los problemas de optimización requieren ser minimizados (costos, esfuerzo, distancia, etc.), pero en términos de calidad, es comúnmente asociado con maximización, ya que un valor más alto supone un mejor rendimiento. Sin embargo, pasar de un problema de minimización a maximización y viceversa es una tarea simple, permitiendo poder abordar cualquier problema de optimización con EA.

4.1.4 Mecanismo para la selección de padres

Los EA implementan muchas técnicas de selección, por ejemplo, torneo y clasificación. La “selección por torneo” opera eligiendo aleatoriamente un número q de individuos de la población generacional y seleccionando el “mejor” para sobrevivir en la próxima generación. Los torneos binarios ($q = 2$) son

probablemente los más comunes. La “selección por clasificación” asigna probabilidades de selección únicamente al rango de un individuo, ignorando los valores absolutos de aptitud. Otras dos técnicas de selección comunes en detalle son las estrategias de selección (μ + λ) y (μ, λ), donde μ representa el número de soluciones padre y λ el número de hijos. El primero selecciona a los mejores μ individuos de entre padres e hijos, el segundo selecciona solo a los individuos μ de la población hija (Goldberg y Deb, 1991).

La “selección por torneo” y la “selección por clasificación” son los mecanismos más populares implementados en EA. Son implementados de manera probabilística, los mejores individuos tienen una probabilidad mayor de ser elegidos como padre, a diferencia de los individuos con bajo rendimiento. Sin embargo, los individuos con bajo rendimiento tienen la oportunidad (pequeña en la mayoría de casos) de ser seleccionados, de no ser así, la diversidad en la población se vería reducida y la búsqueda se convertiría en un método avaro que solo se evoluciona en torno a la primera mejora encontrada, lo que puede ocasionar estancamientos en óptimos locales, y provocar la disminución de la posibilidad de lograr la solución óptima.

4.1.5 Operador genético cruzamiento

El EVOP cruzamiento combina la información genética de dos o más individuos (padres) en uno o más descendientes (hijos). Por lo general, estos métodos están restringidos al uso de dos padres, aunque en algunos casos el uso de múltiples padres ha traído efectos positivos para el algoritmo (Eiben, 2003). Con el proceso de cruzamiento se busca distribuir los genes “buenos” de cada padre en los nuevos individuos, combinando las mejores características de cada padre, pero, esto no siempre se logra pues se realiza de manera probabilística. Para cada representación existen varias metodologías de cruzamiento, algunas realizan segmentación y combinación del material genético de los padres, otras ejecutan operaciones aritméticas que modifican la metodología para crear el nuevo genotipo a partir de las semillas utilizadas.

Un ejemplo del EVOP cruzamiento, es el método “cruzamiento en un punto” (“*one-point crossover*” o “single-point crossover”) que fue propuesto por Holland (1992) y lo podemos describir así: primero calcula un número aleatorio (ρ) en un rango entre $[1, l-1]$, donde l representa el tamaño de la estructura de datos de la codificación (cromosoma), que después utiliza como punto de cruce, dividiendo

los dos padres, cabeza antes de ρ y cola después de ρ, los hijos entonces son la unión de la cabeza de un padre con la cola del otro. Este método se puede generalizar en "cruzamiento de n −puntos" ("n −*point crossover*"): se divide el individuo en más de dos segmentos y se ensambla con material genético de cada padre alternando los segmentos para formar la descendencia; se selecciona un número n de puntos de corte como se ilustra en la **Figura 8**.

El método "cruzamiento uniforme" ("*uniform crossover*"), propone tratar cada gen de forma independiente y realizar una selección para cada hijo que determina el valor del alelo(s) para cada gen, haciendo una elección aleatoria que determina qué padre hereda este valor; en otras palabras, se define un valor para ρ entre 0 y 1 (comúnmente 0,5) luego de manera aleatoria se obtiene otro valor entre 0 y 1, el cual puede ser mayor a ρ indicando que debe tomar el material genético del padre 1 o del padre 2 para el caso que sea menor o igual a ρ.

En el esquema de cruzamiento uniforme medio (HUX, *Half Uniform Crossover*), exactamente la mitad de los bits que son diferentes se intercambian. Por esto es necesario calcular la distancia de Hamming (número de bits diferentes). Este número se divide entre dos, y el número resultante es la cantidad de bits diferentes que tiene que ser intercambiada entre los parentales.

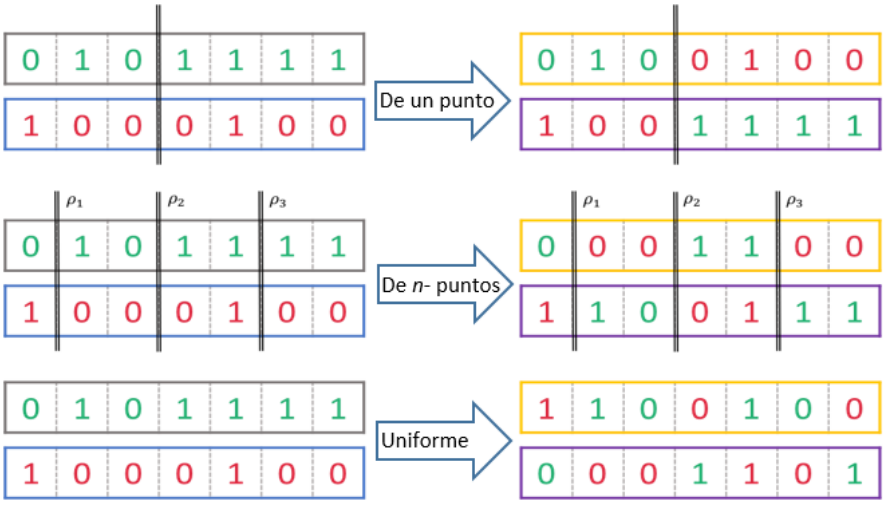

Figura 8. Ejemplos de métodos de cruzamiento. Arriba: "*Cruzamiento de un punto*" para un $\rho = 3$; Centro: "cruzamiento de n −*puntos*" para un $n = 3, \rho = [1,3,5]$; Abajo: "cruzamiento uniforme" para un arreglo de números aleatorios $= [0.1, 0.7, 0.8, 0.2, 0.9, 0.8, 0.2]$ *y* $\rho = 0.5$, cada posición del arreglo se compara con ρ, si es mayor a ρ se usará el material genético del padre 1, y si es menor o igual a ρ se usará el del padre 2.

4.1.6 Operador genético mutación

La mutación es un operador genético utilizado para mantener la diversidad genética de una generación de una población de cromosomas de EA a la siguiente. La mutación altera uno o más valores de genes en un cromosoma desde su estado inicial, o sea, es un operador unario que modifica directamente el material genético de un hijo. En mutación, la solución puede cambiar completamente de la solución anterior. Por lo tanto, un EA puede llegar a una mejor solución mediante el uso de la mutación. La mutación ocurre durante la evolución de acuerdo con una probabilidad de mutación definida por el usuario. Esta probabilidad debe establecerse baja, pues si se establece demasiado alta, la búsqueda se convertirá en una búsqueda aleatoria primitiva.

Algunos ejemplos de métodos de mutación se muestran en la **Figura 9** y son descritos a continuación. En la "mutación de volteo de un bit" ("*bit flip mutation*"), seleccionamos uno o más bits aleatorios y los volteamos, se usa para codificaciones binarias. La "mutación restablecimiento aleatorio" ("*random resetting mutation*") es una extensión del cambio de bit para la representación de enteros, en la cual un valor aleatorio del conjunto de valores permisibles se asigna a un gen elegido aleatoriamente. En la "mutación de intercambio" ("*swap mutation*"), seleccionamos dos posiciones en el cromosoma al azar e intercambiamos los valores, es común en las codificaciones basadas en permutación. En la "mutación de revolver" ("*scramble mutation"*) de todo el cromosoma, se elige un subconjunto de genes y sus valores se revuelven o barajan aleatoriamente, también es popular entre las representaciones de permutación. En la "mutación de inversión" ("*inversion mutation*"), seleccionamos un subconjunto de genes e invertimos la cadena completa en el subconjunto.

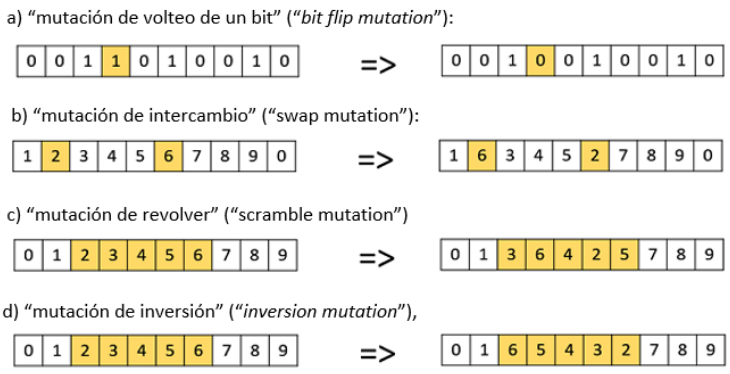

Figura 9. Ejemplos de métodos de mutación.

4.1.7 Mecanismo de selección de los sobrevivientes

El mecanismo de selección de sobrevivientes determina qué individuos serán expulsados y cuáles se mantendrán en la próxima generación. Es crucial ya que debe garantizar que los individuos más aptos no sean expulsados de la población, mientras que al mismo tiempo se debe mantener la diversidad en la población. Es común que muchos EA empleen el elitismo, o sea, que el miembro actual más apto de la población siempre se propaga a la próxima generación.

La política más fácil de selección de sobrevivientes es expulsar a miembros aleatorios de la población, pero este enfoque con frecuencia tiene problemas de convergencia. Las estrategias de ranking o torneo son ampliamente utilizadas para decidir qué individuos deben dejar la población y dar paso al nuevo material genético que contienen los hijos. En un caso en el que el número de hijos es más de la mitad de la antigua población, es necesario eliminar un gran número de individuos que pueden ser seleccionados por edad, bajo desempeño o de manera aleatoria. Cuando se generan dos hijos por generación, se utiliza el término reemplazo, pues mediante torneo binario se define quien continua o no formando parte de la población a evolucionar.

4.2 Algoritmos evolutivos multi-objetivo (MOEA)

Tanto los investigadores como los profesionales de la ciencia, la ingeniería, el gobierno y la industria ciertamente tienen un gran interés en conocer las técnicas más avanzadas de optimización multi-objetivo. Para los investigadores, este es el procedimiento normal para alcanzar contribuciones algorítmicas nuevas y originales. Para los profesionales, este conocimiento les permite elegir el (los) algoritmo(s) más apropiado(s) para su aplicación específica en el dominio de un MOOP. Se desea que un MOEA genere soluciones a un MOOP en el Frente de Pareto, que proporcionen una compensación de rendimiento (eficiencia, efectividad) para objetivos específicos del modelo del sistema (costo / beneficio, restricciones, etc.) que pueden entrar en conflicto. Por ejemplo, el problema clásico multi-objetivo de la mochila: ganancia vs peso, representa un vector de dos objetivos. Maximizar un objetivo, como la ganancia, generalmente no optimiza otro, como el peso (Coello et al., 2007).

Dado que los algoritmos evolutivos y los MOEA en particular pueden codificar soluciones individuales en numerosas representaciones directas (estructuras de datos de cromosomas), así como calcular directamente los valores objetivos asociados, tienen una ventaja considerable sobre las técnicas de búsqueda MOOP tradicionales. Es decir, las técnicas tradicionales pueden imponer restricciones o mapeos complejos en el dominio del problema o en el modelo matemático del dominio del algoritmo para resolver el problema. Por supuesto, el Teorema de No Free Lunch (NFL) implica que un MOEA no es una técnica de solución robusta universal para todas los MOOP (Wolpert y Macready, 1997). Pero, los MOEA generalmente pueden guiarse fácilmente por la información del dominio del problema, sin tener que modificar el modelo del dominio del problema para usarlo con los MOEA. Luego, el proceso de búsqueda es más fácil de desarrollar, comprender y probar en su forma nativa para una aplicación determinada.

Lograr el frente exacto de Pareto de un problema arbitrario suele ser bastante difícil. Sin embargo, las aproximaciones razonablemente buenas del Frente de Pareto ($\mathcal{P}_F$) son generalmente aceptables dentro de un tiempo computacional limitado. Los MOEA, por definición, intentan encontrar estos frentes de Pareto aceptables pero aproximados y las soluciones óptimas de Pareto dentro de alguna medida de error. La primera implementación de un MOEA fue desarrollada por Schaffer (1984), que propuso el algoritmo "*Vector Evaluated Genetic Algorithm"* (VEGA), el cual es una extensión del GA simple para acomodar las medidas de aptitud de valor vectorial. En VEGA el paso de selección se modificó de modo que, en cada generación, se generó una serie de subpoblaciones realizando una selección proporcional de acuerdo con cada función objetivo. Luego se mezclan para obtener una nueva población o subgrupo de individuos a los que se aplican los operadores de cruzamiento y mutación. La siguiente lista histórica (aunque incompleta) de algoritmos de EA para resolver MOOP refleja diferentes marcos algorítmicos, así como la función de aptitud y las representaciones cromosómicas:

- Vector Evaluated Genetic Algorithm (VEGA) (Schaffer, 1984)
- Lexicographic Ordering Genetic Algorithm (Fourman, 1985)
- Vector Optimized Evolution Strategy (VOES) (Kursawe, 1990)
- Weight-Based Genetic Algorithm (WBGA) (Hajela y Lin, 1992)

- Multiple Objective Genetic Algorithm (MOGA) (Fonseca y Fleming, 1993)
- Niched Pareto Genetic Algorithm (NPGA, NPGA 2) (rey Horn et al., 1994), (Erickson et al., 2001)
- Non-dominated Sorting Genetic Algorithm (NSGA, NSGA-II) (Srinivas y Deb, 1994; Deb et al., 2002)
- Distance-based Pareto Genetic Algorithm (DPGA) (Osyczka y Kundu, 1995)
- Thermodynamical Genetic Algorithm (TDGA) (Kita et al., 1996)
- Strength Pareto Evolutionary Algorithm (SPEA, SPEA2) (Zitzler y Thiele, 1999; Zitzler et al., 2001)
- Multi-Objective Messy Genetic Algorithm (MOMGA-I, II, III) (Van Veldhuizen, 1999; Zydallis et al., 2001; Day y Lamont, 2005)
- Pareto Archived Evolution Strategy (PAES) (Knowles y Corne, 2000)
- Pareto Envelope-based Selection Algorithm (PESA, PESA II) (Corne et al., 2000; Corne et al., 2001)
- Micro GA-MOEA (µGA, µGA2) (Coello y Pulido, 2001; Pulido y Coello, 2003)
- Multi-Objective Bayesian Optimization Algorithm (mBOA) (Laumanns y Ocenasek, 2002)
- Multiobjective evolutionary algorithm based on decomposition (MOEA/D) (Zhang y Li, 2007)
- Multiobjective selection based on dominated hypervolume (SMS-EMOA) (Beume et al., 2007)
- Multi-objective Optimization Cellular Genetic Algorithm (MOCell) (Nebro et al., 2009)
- Algorithm for fast hypervolume-based many-objective optimization (HypE) (Bader y Zitzler, 2011)

4.2.1 Clasificación de los MOEA

Varios investigadores clasifican los MOEA en dos grupos o generaciones: la primera integrada por aquellos algoritmos basados en el concepto de dominancia de Pareto y no elitistas (por ejemplo: MOGA, NSGA y NPGA), y la segunda basados en el concepto de dominancia de Pareto y elitistas (por ejemplo: SPEA, SPEA2, NSGA-II, PAES, PESA, PESA-II, MOMGA, MOMGA-II, µGA, µGA2).

La idea de usar asignación de aptitud basada en el concepto de dominancia de Pareto, que caracteriza a los algoritmos de primera generación, fue propuesta por Goldberg (1989) con la adopción de una selección que usara una jerarquización de soluciones, basada en no dominancia a fin de mover una población hacia el frente de Pareto en un MOOP. La idea fundamental es seleccionar a los individuos no dominados con respecto a la población actual y asignarles la jerarquía y la aptitud más elevada. Posteriormente se eliminan temporalmente a estos individuos de la competencia y se re-jerarquiza la población restante. El proceso se repite hasta haber jerarquizado toda la población. Goldberg sugirió también el uso de nichos o alguna técnica similar para mantener diversidad. Esto es necesario debido a que, por ruido estocástico, los algoritmos evolutivos tienden a generar hacia una solución única independientemente de la distribución inicial de la población.

Los algoritmos de segunda generación, se caracterizan por la utilización del elitismo, que proporciona un rendimiento no degradante, porque se mantienen a los individuos con mejor rendimiento durante cada recambio generacional. Los dos esquemas principales de elitismo son:

a. Utilizando un archivo externo (acotado o no acotado) donde se mantienen las soluciones no dominadas global. Se le conoce como población secundaria y se implementa mediante una estructura de datos en memoria donde se almacenan las soluciones no dominadas obtenidas por un MOEA. Ejemplos de algoritmos que lo utilizan: SPEA y SPEA2

b. Utilizando un esquema de selección `+' en el cual se realiza la unión de la población de padres y la de hijos, reteniendo a la mejor mitad usando un ordenamiento total (sin empates). Ejemplos de algoritmos que lo utilizan: NSGA-II y SMS-EMOA.

En los últimos años, la tendencia es clasificar los MOEA en las siguientes categorías (Coello, 2018):

1. Los basados en el concepto de dominancia de Pareto. Por ejemplo, en este grupo se encuentran: SPEA, SPEA2, NSGA, NSGA-II, PESA, PESA II.

2. Los que utilizan métodos por descomposición. Por ejemplo, MOEA/D, el cual divide un MOOP en varios problemas con un único objetivo y que son resueltos simultáneamente.

3. Los que hacen uso de algún indicador. Por ejemplo, SMS-EMOA y HypE, que utilizan la métrica hipervolumen como valor de aptitud para los individuos en un EA.

En los sistemas de almacenamiento en la nube, el rendimiento y la calidad son los requisitos más importantes. Para determinar que algoritmo utilizar para la optimización multi-objetivo del sistema de almacenamiento en la nube AR-RRNS con tres objetivos en conflicto, realizaremos un estudio experimental incluyen tres algoritmos que han demostrado ser muy competitivos: a) *Strength Pareto Evolutionary Algorithm 2* (SPEA2) (Zitzler et al., 2001); b) *Non-dominated Sorting Genetic Algorithm II (NSGA-II)* (Deb et al., 2002); c) *Multi-objective Optimization Cellular Genetic Algorithm* (MOCell) (Nebro et al., 2009). El algoritmo SPEA2 ha sido aplicado en la resolución de muchos problemas de optimización multi-objetivo, es utilizado como referencia para comparar MOEA de nuevo diseño y cuenta con más de 6800 citas en la literatura científica. NSGA-II, que se ha convertido en el MOEA más pupular con más de 28300 citas y variadas aplicaciones en muchas áreas de investigación científica. El algoritmo MOCell tiene más de 300 citas científicas, se comparó con NSGA-II y SPEA2, utilizando 21 problemas de prueba, y los resultados de los indicadores de calidad revelan que MOCell es un algoritmo altamente competitivo considerando las métricas de convergencia, hipervolumen y diversidad. La **Tabla 3** muestra un resumen de las principales características de dichos algoritmos, los cuales serán descritos a profundidad en las próximas secciones.

Tabla 3. Características principales de los algoritmos SPEA2, NSGA-II y MOCell.

Algoritmo	**SPEA2**	**NSGA-II**	**MOCell**
Asignación de aptitud	Fuerza de dominadores	Clasificación basada en ordenamiento de no dominación	Clasificación basada en ordenamiento de no dominación
Mecanismo de diversidad	Densidad basada en el *k-ésimo* vecino más cercano	Distancia de hacinamiento (*Crowding distance*)	Distancia de hacinamiento basada en células
Elitismo	Sí	Sí	Sí
Población externa	Sí	No	Sí
Ventajas	SPEA mejorado, se asegura de preservar los puntos extremos	Parámetro único (N), bien probado, eficiente	Buen rendimiento, convergencia rápida
Desventajas	Cálculo de densidad y aptitud computacionalmente costoso	La distancia de hacinamiento funciona solo en el espacio objetivo	no popular

4.2.2 Strength Pareto Evolutionary Algorithm 2 (SPEA2)

Zitzler y Thiele (1999) introdujeron la primera versión del algoritmo SPEA, como un método que integra diferentes MOEA, buscando combinar lo mejor de cada uno de ellos. SPEA utiliza un archivo externo que contiene soluciones no dominadas encontradas previamente (el llamado conjunto no dominado externo). En cada generación, los individuos no dominados se copian en el conjunto externo no dominado. Para cada individuo en este conjunto externo, se calcula un valor de "fortaleza", que es proporcional al número de soluciones a las que domina un determinado individuo.

En SPEA, la aptitud de cada miembro de la población actual se calcula de acuerdo con las fortalezas de todas las soluciones externas no dominadas que lo dominan. El proceso de asignación de aptitud de SPEA considera la cercanía al verdadero frente de Pareto e incluso la distribución de soluciones al mismo tiempo. Por lo tanto, en lugar de usar nichos basados en la distancia, el dominio de Pareto se utiliza para garantizar que las soluciones se distribuyan correctamente a lo largo del frente de Pareto. Aunque este enfoque no requiere un radio de nicho, su efectividad depende del tamaño del conjunto externo no

dominado. De hecho, dado que el conjunto externo no dominado participa en el proceso de selección de SPEA, si su tamaño crece demasiado, podría reducir la presión de selección, lo que ralentizaría la búsqueda. Debido a esto, los autores decidieron adoptar una técnica que elimina el contenido del conjunto externo no dominado para que su tamaño permanezca por debajo de cierto umbral. El enfoque adoptado para este bien fue una técnica de agrupamiento llamada método de enlace promedio (Coello et al., 2007).

La versión revisada de SPEA (llamada SPEA2) fue propuesta por Zitzler et al. (2001) y tiene tres diferencias principales con respecto a SPEA: (1) incorpora una estrategia de grano fino para asignar la aptitud, la cual tiene en cuenta para cada individuo el número de individuos que lo dominan y el número de individuos a los que domina; (2) utiliza una técnica de estimación de densidad de vecinos más cercanos que guía la búsqueda de manera más eficiente, y (3) tiene un método mejorado de truncamiento de archivo que garantiza la preservación de soluciones de frontera (ver **Figura 10**).

A continuación, se listan algunas de las aplicaciones del algoritmo SPEA2 en varias áreas de la ciencia: Bleuler et al. (2001) para el control del tamaño del código y optimizar el rendimiento en programación genética; Naujoks et al. (2002) para el diseño de aeronaves; García et al. (2011) en la optimización de portafolios de inversión; Sofianopoulos y Tambouratzis (2011) en la optimización de un sistema de traducción basado en reconocimiento de patrones; Adham et al. (2015) en la optimización del rendimiento de un disipador de calor de micro-canal.

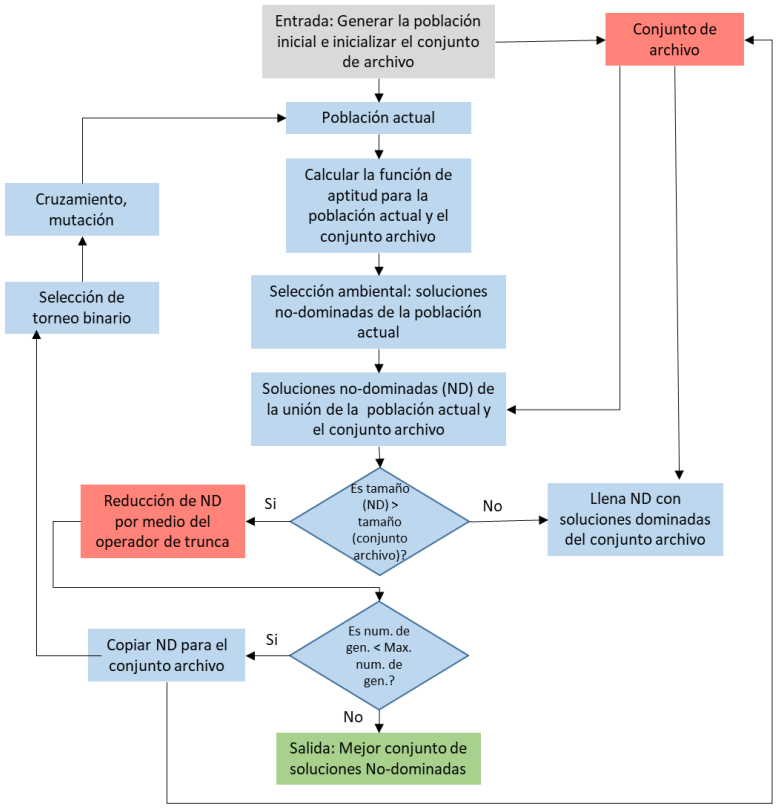

Figura 10. Diagrama de flujo del algoritmo genético SPEA2.

4.2.3 Non-dominated Sorting Genetic Algorithm II (NSGA-II)

En 1994 Kalyanmoy Deb y uno de sus estudiantes de posgrado propusieron la primera versión del Non-dominated Sorting Genetic Algorithm (NSGA) (Srinivas y Deb, 1994). Unos años después Deb et al. (2002) hicieron la propuesta de una versión mejorada: Non-dominated Sorting Genetic Algorithm II (NSGA-II). NSGA-II permite encontrar múltiples soluciones óptimas de Pareto en un MOOP y tiene las siguientes tres características:

1. Utiliza un principio elitista, para lograr mayor eficiencia computacional que el NSGA.

2. Utiliza un mecanismo explícito de preservación de la diversidad, conocido como “crowding distance” o distancia de hacinamiento.

3. Enfatiza las soluciones no dominadas, utilizando el concepto de dominancia por profundidad: este método distribuye los individuos en "frentes" de dominancia, para después ordenarlos de forma tal que el frente uno contiene los mejores individuos, que dominan al resto de la población.

Para cada generación el algoritmo combina la población actual (P_i) de tamaño N con la descendencia (D_i) resultante para cada iteración i (M número de hijos), obtenidos con los operadores genéticos tradicionales (selección, cruzamiento y mutación). A continuación se construye una nueva población (Q_i) de tamaño $M + N$ que es ordenada en "frentes" no dominados, después en cada frente se calcula la distancia de hacinamiento con el objetivo de clasificar a los individuos de un mismo frente en una posición; determinando el aporte de cada individuo a la diversidad del frente. De esta forma, tenemos cada individuo con un vector de k valores de aptitud (para un MOOP de k objetivos), un frente asociado y un valor de distancia de hacinamiento. Finalmente, se construye una nueva población (P_{i+1}) seleccionando los individuos de mejor a peor, es decir, los que pertenecen a los mejores frentes y los que generan mayor diversidad, como se muestra en la **Figura 11**.

La distancia de hacinamiento es una medida del espacio objetivo cerca de una solución u que no es ocupada por otra solución, para calcularla, se halla el perímetro de un rectángulo construido en el espacio objetivo utilizando los vecinos inmediatos ($u - 1$ y $u + 1$) como diagonal.

NSGA-II es muy utilizado en la resolución de MOOP con dos o tres objetivos en conflicto, por ejemplo: Soyel et al. (2011) en la selección de características para reconocimiento de expresiones faciales; Deb et al. (2011) para la optimización de portafolios de inversión; Jemai et al. (2012) para el problema de enrutamiento del vehículo verde (GVRP , *Green Vehicle Routing Problem*); Khishtandar y Zandieh (2017) en la optimización de sistemas de inventarios. En el caso de los problemas de optimización con cuatro o más objetivos en conflicto, se recomienda usar el algoritmo denominado NSGA-III que fue propuesto por Deb y Jain (2013), diseñado específicamente para problemas de optimización de muchos objetivos (MaOPs*, Many-objective optimization problems*).

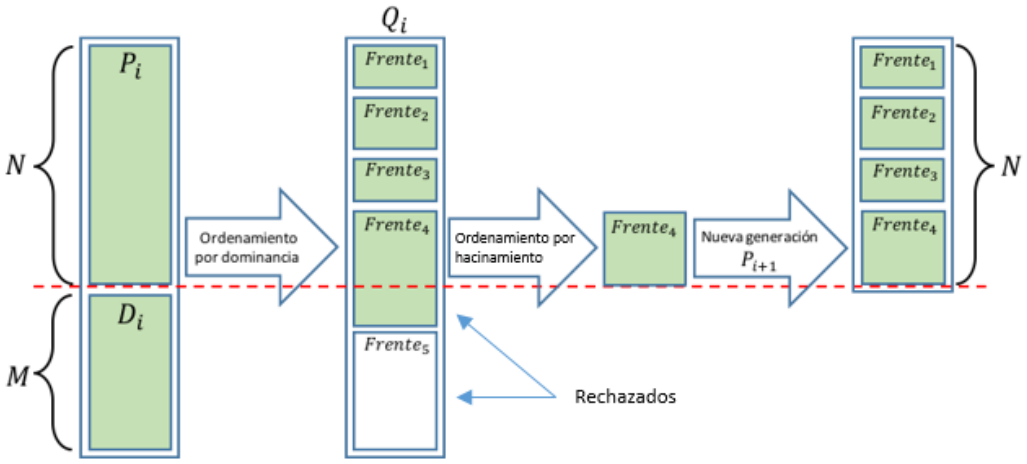

Figura 11. Pasos que ejecuta el algoritmo NSGA-II para crear una nueva generación.

4.2.4 MultiObjective Cellular genetic algorithm (MOCell)

Nebro et al. (2009) propusieron el algoritmo "MultiObjective Cellular genetic algorithm" (MOCell) que ha demostrado ser muy eficiente en la resolución de MOOP de dos y tres objetivos. Este algoritmo genético celular se caracteriza por distribuir los individuos de la población en una malla, después asigna un vecindario a cada individuo (el tamaño del vecindario es configurable en base al problema). Emplea una malla de tipo toroide para asegurar que todos los individuos tengan igual cantidad de vecinos. En este algoritmo el tipo de distribución celular de la población es para restringir la cantidad de individuos que pueden interactuar entre sí. Las restricciones de este tipo promueven la exploración del espacio de búsqueda, de manera que se transmiten genes entre vecindarios sin enfocarse solamente en el mejor individuo de cada subgrupo, conservando una alta diversidad por la difusión "lenta" del material genético. Además, como los individuos interactúan sólo con un reducido grupo de vecinos, se busca fortalecer la explotación en cada vecindario, utilizando las técnicas de clasificación y distancia de hacinamiento popularizadas por el algoritmo NSGA-II y así hace uso del elitismo al aplicar los operadores genéticos.

MOCell crea un conjunto ($\mathcal{P}_{ND}$) de N individuos no dominados (la cardinalidad del conjunto condiciona el nivel de elitismo en el algoritmo) ordenados por distancia de hacinamiento ("crowding distance"), con los que se retroalimenta la población. El algoritmo no hace uso del concepto de recambio

generacional de igual forma que otros EA, pues requiere revisar individuo por individuo para finalizar una generación de la población. Esto implica que en cada evaluación de un individuo se realiza un torneo entre los vecinos para decidir los padres a seleccionar, después se aplica cruzamiento y mutación a los hijos resultantes. Al concluir este proceso se realiza un ordenamiento entre los hijos, los padres seleccionados y el individuo que está siendo evaluado, para definir quién ocupará este espacio en la población (el de mejor rendimiento) y continuar con el siguiente individuo. Al finalizar la revisión de todos los individuos, de manera aleatoria se seleccionan N individuos, los cuales serán intercambiados por aquellos que están ubicados en el conjunto $\mathcal{P}_{ND}$, como se muestra en la **Figura 12**.

El algoritmo MOCell tiene más de 300 citas científicas, a continuación, se listan algunas de sus aplicaciones en varias áreas de la ciencia: Durillo et al. (2009) demostraron que MOCell es muy eficiente para resolver el MOOP conocido como problema de la próxima versión (NRP, *Next Release Problem*) en el ámbito del desarrollo de software; Zhang y Hansen (2009) combinaron las soluciones de los algoritmos genéticos NSGA-II y MOCell para resolver el MOOP de planificación de autogestión en un middleware de servicio generalizado; Guzek et al. (2014) demostraron que MOCell fue el algoritmo de mejor desempeño para resolver el problema de calendarización multi-objetivo de aplicaciones restringidas de precedencia en un sistema informático distribuido, considerando dos objetivos: duración del cronograma y minimización del consumo de energía.

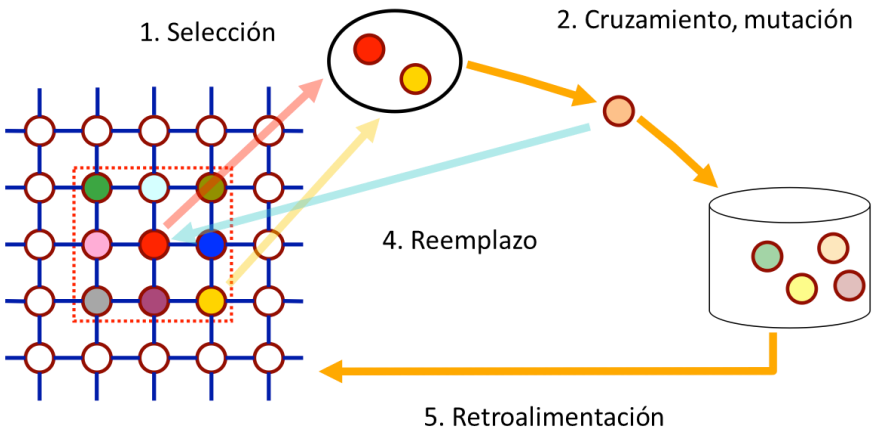

Figura 12. Esquema de funcionamiento del algoritmo genético celular MOCell.

4.3 Indicadores de calidad (métricas de rendimiento)

El objetivo principal de los experimentos con MOEA es comparar algoritmos en términos de efectividad y eficiencia con respecto a un problema (o a un conjunto de problemas de prueba) mediante el uso de métricas apropiadas. Dichas métricas incluyen indicadores de calidad que intentan resumir los resultados experimentales comparativos. Varios investigadores han trabajado en el diseño de mecanismos que permiten comparar los resultados entregados por dos algoritmos y determinar cuál presento un mejor rendimiento. Zitzler et al. (2000) proponen tres metas para los MOEA:

- Minimizar la distancia entre el vector de soluciones no dominadas y el frente de Pareto ($\mathcal{P}_F$), o sea, deben buscar estar lo más cerca posible de la solución óptima (convergencia).

- Lograr una distribución uniforme de las soluciones del conjunto no dominado en el espacio objetivo (diversidad).

- Maximizar la cardinalidad del conjunto solución (cantidad de soluciones no dominadas).

Riquelme et al. (2015) presentaron una revisión y análisis de 54 métricas de optimización multi-objetivo que se han empleado en la literatura especializada, discutiendo el uso, la tendencia y las ventajas/desventajas. El proceso de revisión realizado en dicho trabajo indica que los indicadores más utilizados son: hipervolumen, distancia generacional, indicador de épsilon aditivo y distancia generacional invertida.

Una métrica de rendimiento es una función I de orden m, $I: \mathbb{P}^m \rightarrow \mathbb{R}$, que asigna para cada conjunto $\mathbb{P} = [\mathcal{P}_1, \dots, \mathcal{P}_m]$ de m conjuntos de aproximación, un valor real $I(\mathcal{P}_1, \dots, \mathcal{P}_m)$ (Zitzler et al., 2002). Hay dos formas principales de clasificar las métricas. El primer criterio de clasificación considera los aspectos que las métricas miden al evaluar los conjuntos de aproximación en $\mathbb{P}$. Considerando un conjunto de aproximación $\mathcal{P}$, las métricas se pueden agrupar como (Okabe et al., 2003):

- Métricas de cardinalidad: permiten comparar la cardinalidad de un conjunto solución $\mathbb{P}$ (la cantidad de soluciones $\mathcal{P}$ que existen).

- Métricas de exactitud: permiten evaluar el nivel de convergencia de un $\mathcal{P}$, o sea, indica la distancia que separa $\mathcal{P}$ de $\mathcal{P}_F$ si este último es conocido, de lo contrario se usará un $\mathcal{P}_R$ (frecuentemente calculado de forma experimental) como referencia (por ejemplo, "hipervolumen" y "cubrimiento de dos conjuntos").

- Métricas de diversidad: permiten analizar la distribución y extensión de las soluciones en $\mathcal{P}$, estos dos términos se diferencia en que, la distribución indica la distancia relativa entre las soluciones no dominadas, mientras que la extensión se refiere al rango de valores que cubre una solución en $\mathcal{P}$ (por ejemplo, "espaciado", "dispersión", "hipervolumen").

El segundo criterio de clasificación tiene en cuenta el número m de conjuntos de aproximación que evaluará la métrica. Se han utilizado dos tipos de métricas en la literatura (Riquelme et al., 2015):

- Métrica unaria: se dice que la métrica es unaria si recibe como parámetro solo un conjunto de aproximación $\mathcal{P}$ para ser evaluado.

- Métrica binaria: se dice que la métrica es binaria si recibe como parámetro dos conjuntos de aproximación, $\mathcal{P}_1$ y $\mathcal{P}_2$.

En este trabajo utilizamos tres indicadores de calidad muy populares en la literatura especializada en optimización multi-objetivo, ellos son: distancia generacional invertida (IGD, *Inverted Generational Distance*), indicador de épsilon aditivo (EP, *Additive Epsilon Indicator*) e hipervolumen (HV, *Hypervolume*). A continuación, realizaremos un breve resumen de algunos conceptos de optimización multi-objetivo para luego explicar dichos indicadores de calidad.

Consideremos un MOOP (X, f), donde X es el espacio de solución y $f = (f_1, \dots, f_i, \dots, f_d)$ es un vector de funciones objetivo tal que f_i se debe minimizar para todo $i \in \{1, \dots, d\}$. Sea $Z = f(X)$ el espacio objetivo, $Z \subseteq \mathbb{R}^d$. Cada solución $x \in X$ está asociada con un vector objetivo $z \in Z$ tal que $z = f(x)$.

Un vector objetivo $z \in Z$ está dominado por un vector objetivo $z' \in Z (z \prec z')$ si y solo si, $\forall i \in \{1, \dots, d\} : z'_i \leq z_i$ y $\exists i \in \{1, \dots, d\} : z'_i \leq z_i$ tal que $z'_i < z_i$. Dos vectores objetivos $z, z' \in Z$ están mutuamente no dominados si y solo si, $z \nprec z'$ y $z' \nprec z$. Un vector objetivo $z^* \in Z$ es Pareto óptimo o no está dominado, si y solo si, no existe un $z \in Z$ tal que $z^* \prec z$ (Liefooghe y Derbel, 2016).

Se pueden formalizar definiciones similares para soluciones $x \in X$ utilizando los vectores objetivos asociados $z \in Z$, como $z = f(x)$. El frente de Pareto $Z^* \subseteq Z$ es el conjunto de vectores objetivos no dominados. El conjunto de Pareto $X^* \subseteq X$ es un conjunto de soluciones que se asigna al frente de Pareto, es decir, $f(X^*) = Z^*$. Uno de los problemas más desafiantes en la optimización multi-objetivo es identificar el conjunto / frente de Pareto, o su buena aproximación para problemas complejos.

Un indicador de calidad (unario) es una función $2^Z \to \mathbb{R}$ que asigna cada conjunto de aproximación a un valor (escalar) que refleja su calidad. Sea $A \subseteq Z$ un conjunto de vectores objetivos mutuamente no dominados (es decir, una aproximación al Frente de Pareto o un conjunto de aproximación), y $R \subseteq Z$ sea un conjunto de referencia (idealmente el frente de Pareto cuando es discreto, es decir, $R = Z^*$). En lo adelante, supondremos que no existe ningún vector en A que domine a un vector en R; es decir, $\forall r \in R, \nexists a \in A$ tal que $r \prec a$. En otras palabras, el conjunto de referencia R domina débilmente cualquier conjunto de aproximación A.

IGD: La distancia generacional invertida (Van Veldhuizen y Lamont, 1998) proporciona la distancia promedio entre cualquier punto desde el conjunto de referencia R y su punto más cercano desde el conjunto de aproximación A, se puede formalizar como:

$$\mathrm{IGD}(A) = \frac{1}{|R|}\sqrt{\sum_{r \in R} \min_{a \in A} ||a - r||_2^2}\,. \tag{21}$$

La distancia euclidiana (norma-L2) en el espacio objetivo generalmente se usa para calcular la distancia. Obviamente, cuanto menor es el valor de IGD, más se acerca el conjunto de aproximación del conjunto de referencia. Un valor indicador de 0 implica $R \subseteq A$. En la **Figura 13** se muestra un ejemplo del cálculo de este indicador.

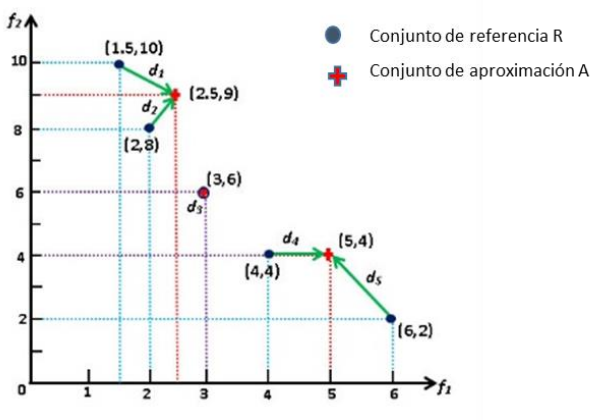

Figura 13. El indicador de calidad Distancia Generacional Invertida (IGD).

EP: El indicador de épsilon aditivo (Zitzler et al., 2002) proporciona el factor mínimo por el cual el conjunto de aproximación debe traducirse en el espacio objetivo para dominar (débilmente) el conjunto de referencia. Se basa en un factor aditivo y se puede describir formalmente de la siguiente forma:

$$\mathrm{EP}(A) = \max_{r \in R} \min_{a \in A} \max_{i \in \{1,\dots,d\}} (a_i - r_i). \tag{22}$$

Cuanto menor sea el valor de EP, más se acercará el conjunto de aproximación al conjunto de referencia. Un valor indicador de 0 implica $R \subseteq A$.

HV: El hipervolumen (Zitzler y Thiele, 1999) proporciona el volumen multidimensional de la porción del espacio objetivo que está débilmente dominado por un conjunto de aproximación. Formalmente lo podemos definir como:

$$\mathrm{HV}(A) = \int_{z^{min}}^{z^{max}} \alpha_A(z) dz\,, \tag{23}$$

$$\text{tal que: } \alpha_A(z) \coloneqq \begin{cases} 1 & si\ \exists a\ \in A\ tal\ que\ z\ \prec a, \\ 0 & de\ otra\ forma. \end{cases}$$

En la práctica, solo se requiere el vector de límite superior $z^{max} \in \mathbb{R}^d$ para calcular el hipervolumen. Este parámetro se llama punto de referencia o W. Un frente de Pareto con un HV más alto que otro podría deberse a dos factores: algunas soluciones en el primer frente dominan las soluciones en el segundo, o las soluciones en el primer frente están mejor distribuidas que en el segundo. Por lo tanto, los algoritmos con valores mayores de HV son deseables, debido a que el valor más alto (valor óptimo) de esta métrica se obtiene al evaluar el frente de Pareto. La principal ventaja del hipervolumen sobre muchos otros indicadores de rendimiento es su propiedad de cumplimiento de Pareto. Por ejemplo, en la **Figura 14** se muestra una comparativa del indicador hipervolumen generado por dos conjuntos de soluciones $\mathcal{P}_1$ y $\mathcal{P}_2$, en un problema de bi-objetivo de minimización, se puede apreciar que el $\mathcal{P}_1$ alcanza mayor valor de hipervolumen y por tanto es una mejor aproximación al frente de Pareto.

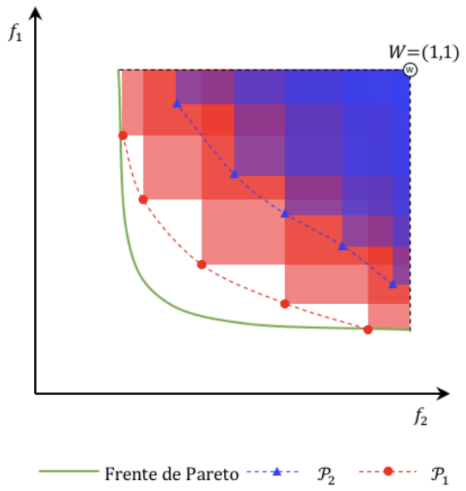

Figura 14. Comparativa del indicador hipervolumen generado por los conjuntos de soluciones $\mathcal{P}_1$ y $\mathcal{P}_2$.

4.4 El framework jMetal para la optimización multi-objetivo

Existen varios marcos de trabajo (*frameworks*) que son utilizados en la optimización multi-objetivo con metaheurísticas. Entre ellos, "jMetal" (Durillo y Nebro, 2011; Nebro et al., 2015) se destaca por las siguientes razones: es un framework implementado en el lenguaje de programación Java, orientado a objetos, de código abierto, basado en patrones de diseño de software, fácil de usar, flexible, extensible, portable, permite ejecución paralela y cuenta con una activa comunidad de desarrolladores.

El framework jMetal es utilizado por muchos investigadores en el área, muestra de ello es que tiene más de 1100 citas científicas. La arquitectura orientada a objetos del framework y las características incluidas, permite a sus usuarios: experimentar con las técnicas clásicas y de vanguardia proporcionadas, desarrollar sus propios algoritmos, resolver sus problemas de optimización, integrarlo en otras herramientas, etc. La versión actual es jMetal 5.6, que incluye las siguientes funcionalidades:

- Algoritmos de optimización multi-objetivo soportados: NSGA-II, SPEA2, PAES, PESA-II, OMOPSO, MOCell, AbYSS, MOEA/D, GDE3, IBEA, SMPSO, SMPSOhv, SMS-EMOA, MOEA/D-STM, MOEA/D-DE, MOCHC, MOMBI, MOMBI-II, NSGA-III, WASF-GA, GWASF-GA, R-NSGA-II, CDG-MOEA, ESPEA, SMSPO/RP.

- Conjunto de problemas de prueba:

    - Familias de problemas: ZDT, DTLZ, WFG, CEC2009, LZ09, GLT, MOP, CEC2018.

    - Problemas clásicos: Kursawe, Fonseca, Schaffer, Viennet2, Viennet3.

    - Problemas restringidos: Srinivas, Tanaka, Osyczka2, Constr_Ex, Golinski, Water, Viennet4.

    - Problemas combinatorios: El problema del vendedor itinerante multi-objetivo (TSP).

    - Problemas académicos: OneMax, OneZeroMax.

- Indicadores de calidad: hipervolumen, dispersión, distancia generacional, distancia generacional invertida, distancia generacional invertida plus, épsilon aditivo.

- Representaciones variables: binario, real, entero, permutación, codificación mixta (real + binario, int + real).

Por todas estas razones, JMetal, se elige como el framework para la optimización multi-objetivo del sistema de almacenamiento AR-RRNS.

Capítulo 5. Metodología y resultados experimentales

En este Capítulo, se define el problema de optimización multi-objetivo AR-RRNS para ser resuelto mediante EMOO, se describirá el proceso de calibración de los parámetros de los GAs, los datos de entrada, los operadores genéticos, la configuración de los algoritmos y la metodología de la evaluación experimental. Luego, se presentan los resultados experimentales junto a una discusión sobre el análisis de estos.

5.1 Definición del MOOP AR-RRNS con enfoque de algoritmos evolutivos

En esta sección se describe la formulación de nuestro problema de optimización multi-objetivo AR-RRNS (planteado en el Capítulo 2), para ser resuelto mediante algoritmos evolutivos con enfoque multi-objetivo. Se plantea la codificación del cromosoma y la función de evaluación de la aptitud.

5.1.1 Representación (codificación)

Para resolver nuestro problema, utilizamos una representación del cromosoma que contiene dos cadenas binarias. La primera de longitud N (cantidad total de nubes), en la cual si $c_j = 1$, la nube j se utiliza para almacenar información, además el número de elementos no-cero determina el parámetro n de nuestra configuración (k, n) y su valor se restringe al rango $2 \leq n \leq N$. La segunda cadena binaria tiene una longitud de $\log_2 N$, representa el parámetro k y su valor se restringe al rango $2 \leq k \leq n$. Por ejemplo, el cromosoma en la **Figura 15** representa una solución $(k, n) = (3, 7)$, especificando además que la información se almacena en las nubes $c_2, c_3, c_5, c_7, c_8, c_{10}, c_{11}$.

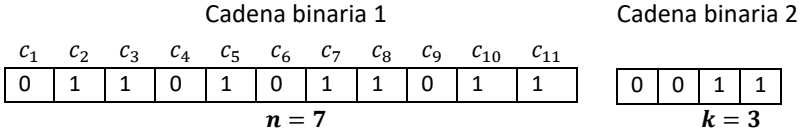

Figura 15. Ejemplo de representación del cromosoma para el MOOP AR-RRNS.

En el análisis del cromosoma propuesto se tuvieron en cuenta los siguientes principios para evaluar una codificación (Gen y Lin, 2007):

1. Espacio: el cromosoma requiere poca cantidad de memoria.

2. Tiempo: baja complejidad temporal de las tareas: evaluar el cromosoma, cruzamiento y mutación.

3. Viabilidad: los cromosomas generados, incluyendo generados por cruzamiento simple (es decir, un cruzamiento de un punto de corte) y mutación, representan soluciones factibles.

4. Singularidad: el mapeo de cromosomas a soluciones (decodificación) utilizado es mapeo 1 a 1, que es muy eficiente.

5. Heredabilidad: la descendencia de un cruzamiento simple representa soluciones que combinan subestructuras de sus soluciones parentales.

6. Localidad: un cromosoma mutado representa una solución similar a la de su padre.

5.1.2 Funciones objetivo y evaluación de aptitud

El MOOP AR-RRNS (ver **Sección 2.3**) es formulado mediante un vector con tres funciones objetivo en conflicto, de la siguiente forma $\mathbb{F} = [P_r(k,n), R, T_{ex}]$, que deben ser minimizadas. La primera función de dicho vector representa la probabilidad de pérdida de información, y se calcula de la siguiente forma: $P_r(k,n) = \sum_{A \in F_{n-k+1}}^{n} \prod_{j \in A} err_j \prod_{j^c \in A^c} (1 - err_{j^c})$, donde el conjunto F_{n-k+1} es el conjunto de todos los posibles $n-k+1$ subconjuntos del conjunto de nubes C, mientras que A^c es el complemento de los subconjuntos A y C, se denota err_j a la probabilidad de falla de la nube c_j. La información se puede perder solo si $n-k+1$ partes tienen error o se perdieron al ser recuperadas de las nubes seleccionadas. La segunda función: R, representa la redundancia que es la razón del tamaño de la información codificada almacenada D_E y el tamaño original de la información D, o sea $R = \frac{D_E}{D}$. La tercera función: T_{ex},

representa el tiempo de extracción, interpretado como que tan rápido se descarga la información de cada una de las nubes y se decodifica para que pueda ser usada, se calcula como la suma del tiempo de decodificación T_D y el tiempo de descarga T_{dow}. Por lo tanto $T_{ex} = T_D + T_{dow}$, donde $T_{dow} = \sum_{i=n-k+1}^{n} \frac{S_{Ei}}{d_i}$, suponiendo que cada uno de los pedazos es descargado de forma secuencial de las nubes. El proceso de descarga termina cuando k pedazos son descargados de forma correcta.

Al realizar un análisis de los resultados experimentales del modelo de almacenamiento AR-RRNS, se obtuvo la relación entre las diferentes configuraciones (k, n) versus: a) el tiempo de decodificación (ver **Figura 16**); b) la redundancia (ver **Figura 17**). Partiendo de dichos datos, se implementó la función para evaluar la aptitud de cada individuo de la población del GA. El **Anexo 1** muestra los resultados experimentales del modelo AR-RRNS.

Las restricciones del modelo para garantizar un umbral de seguridad, son: a) utilizar 2 o más nubes para almacenar fragmentos de información: $n \geq 2$; b) descargar fragmentos de 2 o más nubes para reconstruir la información original: $k \geq 2$; c) el número de nubes donde se almacena la información debe ser mayor o igual al número de nubes necesarias para recuperar la información: $k \leq n$.

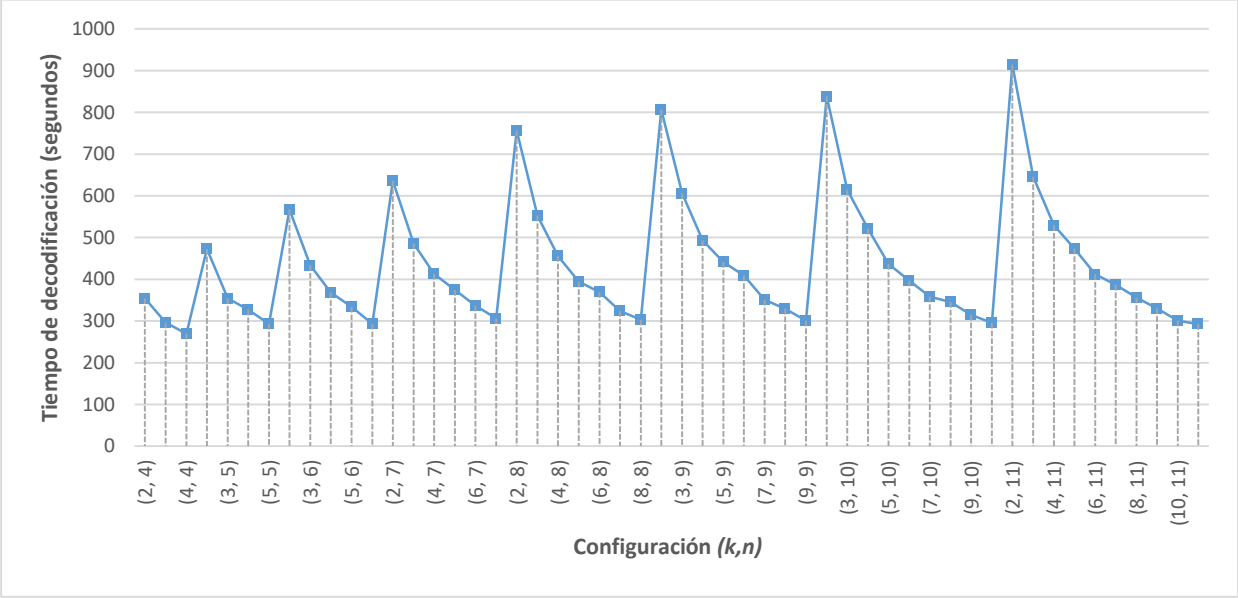

Figura 16. Tiempo de decodificación del modelo AR-RRNS para diferentes configuraciones (k, n).

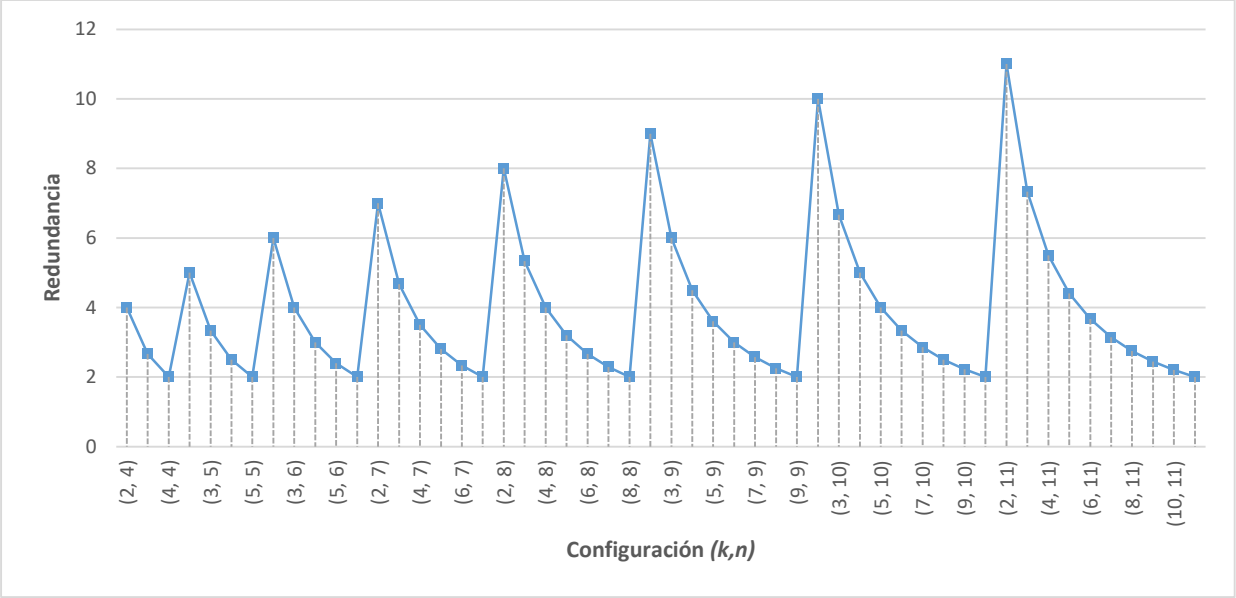

Figura 17. Redundancia del modelo AR-RRNS para diferentes configuraciones (k, n).

5.2 Calibración de parámetros de los GAs

El método utilizado en la calibración de los parámetros de configuración de los algoritmos genéticos seleccionados (SPEA2, NSGA-II y MOCell), consta de los siguientes pasos:

a) Probar todas las instancias producidas con posibles combinaciones de parámetros para cada algoritmo;

b) Obtener la mejor solución para cada caso;

c) Aplicar el Análisis de varianza multifactorial (ANOVA) con un nivel de confianza del 95% para encontrar los parámetros más influyentes, nuestra hipótesis nula H_0: los cambios en los parámetros de configuración de los algoritmos genéticos no tienen influencia en su rendimiento;

d) Establecer los parámetros de los algoritmos basados en valores de parámetros seleccionados;

e) Calcular la diferencia relativa del algoritmo calibrado y otros algoritmos adaptados sobre las mejores soluciones.

En los experimentos, se utilizó la siguiente plataforma de software: Windows 10 Enterprise de 64 bits, jMetal 5.6 y JDK 11.0.1. La plataforma de hardware empleada fue: Dell Precision T3610, Intel Xeon CPU E5-1606 @ 2.80 GHz, 16 GB de RAM DDR3.

Los parámetros configurados para la calibración fueron los siguientes:

- Operadores de cruzamiento: “cruzamiento de un punto” y “cruzamiento *uniforme medio*”;
- Probabilidad de cruzamiento (P_c): 0.6, 0.7, 0.8 y 0.9;
- Probabilidad de mutación (P_m): 0.05, 0.1, 0.2 y 0.3.

Por lo tanto, 2 * 4 * 4 = 32 configuraciones diferentes se consideran para calibrar los algoritmos. Se realizaron 40 ejecuciones independientes por cada configuración. Además, fueron utilizados los siguientes valores de configuración comunes:

- Operador de mutación: “mutación de volteo de un bit”
- Tamaño de población: 100
- Número máximo de evaluaciones: 25,000
- Método de selección: “selección de torneo binario”

La medida de rendimiento de los algoritmos se calculó como el porcentaje de la distancia relativa desde la solución obtenida hasta la mejor:

$$\frac{Heu_{sol} - Best_{sol}}{Best_{sol}} \times 100, \qquad Heu_{sol} = (1 - I_{HV}) + I_{EP} + I_{IGD}. \tag{24}$$

Donde Heu_{sol} es la suma de los indicadores de calidad (EP, HV, IGD), que se normalizan en el rango {0 ... 1}. $Best_{sol}$ es el mejor valor obtenido durante la prueba de todas las combinaciones posibles de parámetros.

Para evaluar la diferencia estadística entre los resultados experimentales y observar cómo los parámetros impactan en la calidad resultante, se aplica la técnica ANOVA, consideramos un nivel de confianza del 95% (p-valor inferior a 0,05). El análisis de varianza se utiliza para determinar los factores que tienen un efecto significativo y los factores más importantes. Los parámetros de los algoritmos genéticos se consideran factores y sus valores como niveles. Los resultados se presentan como un aumento sobre la mejor solución en porcentaje. Los datos se expresan en términos de media y desviación estándar (el valor más bajo es el mejor).

- En la **Figura 18** se muestran los resultados de análisis de los operadores de cruzamiento, donde el "cruzamiento de un punto" ($7.5 \pm_{0.5e+01}$) es mejor que el "cruzamiento uniforme medio" ($14.8 \pm_{0.5e+01}$) con un *p*-valor = 0,015.

- En la **Figura 19** se presentan los resultados del análisis de la probabilidad de cruzamiento, donde el valor 0.9 ($8.1 \pm_{0.3+01}$) es mejor que los valores: 0.6 ($25.0 \pm_{0.2e+01}$), 0.7 ($30.0 \pm_{0.3e+01}$), y 0.8 ($21.5 \pm_{0.2e+01}$) con un *p*-valor = 0,017.

- En la **Figura 20** se observan los resultados del análisis de la probabilidad de mutación, en el cual el valor 0.1 ($11.5 \pm_{0.7e+01}$) es mejor que los valores: 0.05 ($22.3 \pm_{0.6e+01}$), 0.2 ($17.5 \pm_{0.7e+01}$), y 0.3 ($16.2 \pm_{0.8e+01}$) con un *p*-valor = 0,013.

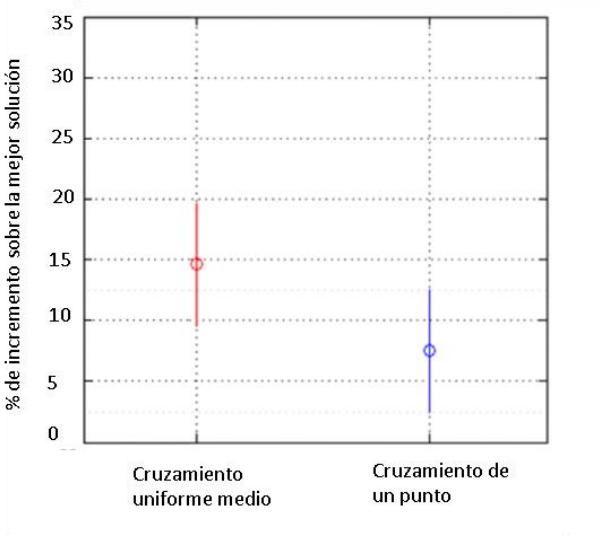

Figura 18. Calibración de parámetros de los GAs. Análisis de los operadores de cruzamiento, presentado como el porcentaje de aumento sobre la mejor solución. El valor más bajo es el mejor.

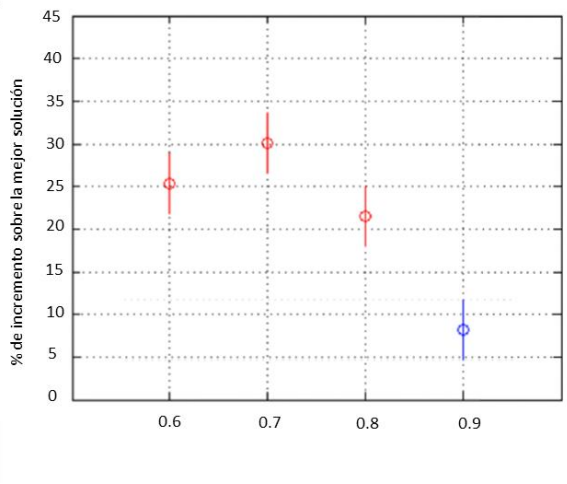

Figura 19. Calibración de parámetros de los GAs. Análisis de la probabilidad de cruzamiento, presentado como el porcentaje de aumento sobre la mejor solución. El valor más bajo es el mejor.

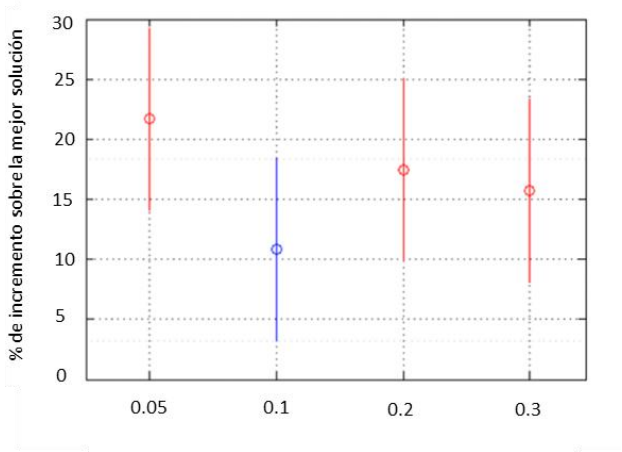

Figura 20. Calibración de parámetros de los GAs. Análisis de la probabilidad de mutación, presentado como el porcentaje de aumento sobre la mejor solución. El valor más bajo es el mejor.

5.3 Configuración experimental

Todos los algoritmos se ejecutan con un máximo de 25,000 evaluaciones y se ajustan con los siguientes parámetros obtenidos durante el paso de calibración:

- Tamaño de la población: 100
- Probabilidad del operador de cruzamiento: $P_c = 0.9$
- Probabilidad del operador de mutación: $P_N = 1/N$ (donde N es el número de nubes. Por ejemplo, para $N = 11$, sería, ≈ 0.1)
- Operador de cruzamiento: “cruzamiento de un punto”
- Operador de mutación: “mutación de volteo de un bit”
- Método de selección: “selección de torneo binario”

Utilizamos datos reales de 11 proveedores de almacenamiento en la nube, obtenidos de los trabajos de Tchernykh et al. (2018) y Lopez-Falcon et al. (2018), que muestran para cada nube los valores mínimos, máximos y promedio de: velocidad de carga / descarga (**Tabla 4**) y probabilidad de falla (**Tabla 5**).

Realizamos un estudio experimental para comparar SPEA2, NSGA-II y MOCell. Se consideran cuatro instancias del problema: AR-RRNS-8 (8 nubes: $c_1, c_2, \dots, c_8$), AR-RRNS-9 (9 nubes: $c_1, c_2, \dots, c_9$), AR-RRNS-10 (10 nubes: $c_1, c_2, \dots, c_{10}$) y AR-RRNS-11 (11 nubes: $c_1, c_2, \dots, c_{11}$). Fueron ejecutadas 50 ejecuciones independientes de cada experimento y registramos la media y la desviación estándar. Como no hay referencia de Pareto Front para nuestro MOOP, utilizamos la funcionalidad jMetal que nos permite calcular el frente de Pareto aproximado, a partir de la combinación de los puntos no dominados de los 3 algoritmos genéticos. Como se explicó en la Sección 2.3, el problema que se aborda en el presente trabajo, para aplicar AR-RRNS, es: encontrar la configuración (k, n) y un subconjunto C' de nubes minimizando los tres objetivos: la probabilidad de pérdida de información ($P_r(k, n)$), la redundancia (R) y el tiempo de extracción (T_{ex}).

Tabla 4. Velocidades de carga / descarga de cada nube.

	Proveedor	**Velocidad de carga u_j (MBps)**			**Velocidad de descarga d_j (MBps)**		
		Min	**Max**	**Promedio**	**Min**	**Max**	**Promedio**
c_1	Google Drive	1.79	3.24	2.98	2.15	3.26	3.06
c_2	OneDrive	0.91	1.70	1.46	1.21	2.41	2.18
c_3	Dropbox	2.59	3.05	2.93	3.07	3.32	3.25
c_4	Box	1.91	3.26	2.55	2.01	3.20	2.62
c_5	Egnyte	1.24	1.93	1.70	2.17	2.36	2.30
c_6	Sharefile	0.11	0.65	0.51	0.72	0.76	0.75
c_7	Salesforce	0.52	0.73	0.64	0.68	0.72	0.71
c_8	Alibaba Cloud	2.32	3.14	2.73	2.54	3.18	2.86
c_9	Amazon Cloud Drive	0.70	1.86	1.28	2.49	3.09	2.79
c_{10}	Apple iCloud	2.05	3.45	2.75	2.01	2.98	2.49
c_{11}	Azure Storage	1.31	3.17	2.24	2.30	3.12	2.71

Tabla 5. Probabilidad de falla de cada nube.

	Proveedor	Probabilidad de falla		
		Min	Max	Promedio
c_1	Google Drive	0.00072679	0.00145361	0.00109019
c_2	One Drive	0.00066020	0.00132043	0.00099030
c_3	DropBox	0.00097032	0.00194065	0.00145548
c_4	Box	0.00179699	0.00359402	0.00269549
c_5	Egnyte	0.00073249	0.00146503	0.00109874
c_6	Sharefile	0.00014269	0.00028542	0.00021404
c_7	Salesforce	0.00061739	0.00123480	0.00092609
c_8	Alibaba Cloud	0.00079897	0.00138793	0.00109345
c_9	Amazon Cloud Drive	0.00018227	0.00098241	0.00058234
c_{10}	Apple iCloud	0.00015664	0.00097632	0.00076648
c_{11}	Azure Storage	0.00039977	0.00087241	0.00063609

Para simular el impacto de la variabilidad de parámetros en un ambiente multi-nube, se realizó un segundo estudio experimental con los 3 GAs, durante 60 simulaciones con distintos valores de velocidad de descarga y probabilidad de falla, de las nubes en cada simulación. Los valores para las nubes se seleccionaron con una distribución uniforme entre el valor mínimo y el máximo, de velocidad de descarga y probabilidad de falla por cada nube. En cada simulación se utilizaron tres instancias del nuestro MOOP: VA-AR-RRNS-9 (9 nubes: $c_1, c_2, \dots, c_9$), VA-AR-RRNS-10 (10 nubes: $c_1, c_2, \dots, c_{10}$) y VA-AR-RRNS-11 (11 nubes: $c_1, c_2, \dots, c_{11}$).

Para evaluar las capacidades de búsqueda de los algoritmos, hemos realizado 40 corridas independientes de cada experimento, y mostramos la media, $\bar{x}$ y la desviación estándar, σ_n, como medidas de ubicación (o tendencia central) y dispersión estadística, respectivamente. Dado que estamos tratando con algoritmos estocásticos para proporcionar resultados confiables, también hemos incluido una fase de prueba al realizar una comparación múltiple de muestras (Hochberg y TAMHANE, 1987). Hemos utilizado la prueba de Wilcoxon para ese propósito. Siempre consideramos un nivel de confianza del 95% (es decir, un nivel de significación del 5% o un *p*-valor (*p-value*) inferior a 0,05) en las pruebas estadísticas.

5.4 Resultados obtenidos y discusión

A continuación, se presentan los resultados del estudio experimental de los algoritmos SPEA2, NSGA-II y MOCell para el modelo de almacenamiento AR-RRNS. Se describen los resultados para cada uno indicadores de calidad explicados en la **Sección 4.3**, en las siguientes tablas, el mejor valor de cada indicador de calidad está marcado por un fondo sombreado.

Considerando el indicador de calidad de Épsilon (EP) (ver **Tabla 6**), MOCell logra los mejores resultados (el valor más bajo con una desviación estándar más baja es mejor), seguido por SPEA2, luego por NSGA-II. Tiene un I_{EP} 77.5% mejor que SPEA2 y 83.3% mejor que NSGA-II.

Tabla 6. Indicador de calidad EP. Media y desviación estándar, el mejor valor está marcado por un fondo sombreado.

Problema	SPEA2 $\overline{x}_{\sigma_n}$	NSGA-II $\overline{x}_{\sigma_n}$	MOCell $\overline{x}_{\sigma_n}$
AR-RRNS-8	$0.00e{+}00 \pm_{0.0e+00}$	$1.34e-03 \pm_{4.2e-04}$	$0.00e{+}00 \pm_{0.0e+00}$
AR-RRNS-9	$5.20e-03 \pm_{1.3e-03}$	$2.71e-03 \pm_{3.7e-04}$	$5.87e-04 \pm_{0.0e+00}$
AR-RRNS-10	$1.42e-02 \pm_{3.2e-03}$	$4.46e-03 \pm_{2.6e-03}$	$2.08e-03 \pm_{4.4e-04}$
AR-RRNS-11	$2.53e-03 \pm_{1.0e-03}$	$3.78e-03 \pm_{5.2e-04}$	$8.98e-04 \pm_{1.0e-04}$

La **Tabla 7** muestra el indicador de hipervolumen (HV), donde el valor más grande con una desviación estándar más baja es mejor. En este indicador MOCell y NSGA-II obtuvieron resultados similares, ambos logrando un valor en I_{HV} 2.0% mejor que SPEA2.

Tabla 7. Indicador de calidad HV. Media y desviación estándar, el mejor valor está marcado por un fondo sombreado.

Problema	SPEA2 $\overline{x}_{\sigma_n}$	NSGA-II $\overline{x}_{\sigma_n}$	MOCell $\overline{x}_{\sigma_n}$
AR-RRNS-8	$9.55e-01 \pm_{0.0e+00}$	$9.55e-01 \pm_{1.2e-06}$	$9.55e-01 \pm_{0.0e+00}$
AR-RRNS-9	$9.51e-01 \pm_{1.2e-03}$	$9.54e-01 \pm_{2.9e-06}$	$9.54e-01 \pm_{5.7e-08}$
AR-RRNS-10	$9.63e-01 \pm_{1.8e-03}$	$9.67e-01 \pm_{9.4e-05}$	$9.67e-01 \pm_{1.0e-06}$
AR-RRNS-11	$9.77e-01 \pm_{1.2e-04}$	$9.77e-01 \pm_{9.3e-06}$	$9.77e-01 \pm_{1.5e-07}$

Considerando el indicador IGD (ver **Tabla 8**), donde el valor más bajo con una desviación estándar más baja es mejor, MOCell supera al resto de los algoritmos, alcanzando un valor de I_{IGD} 78.0% mejor que SPEA2 y 80.3% mejor que NSGA-II.

Podemos concluir que MOCell es mejor que SPEA2 y NSGA-II en todos los casos de prueba, teniendo en cuenta tres indicadores de calidad. Alcanza un buen equilibrio entre convergencia y diversidad.

Tabla 8. Indicador de calidad IGD. Media y desviación estándar, el mejor valor está marcado por un fondo sombreado.

Problema	SPEA2 $\overline{x}_{\sigma_n}$	NSGA-II $\overline{x}_{\sigma_n}$	MOCell $\overline{x}_{\sigma_n}$
AR-RRNS-8	$0.00e+00 \pm_{0.0e+00}$	$3.09e-04 \pm_{5.6e-05}$	$0.00e+00 \pm_{0.0e+00}$
AR-RRNS-9	$1.21e-04 \pm_{1.6e-05}$	$4.03e-04 \pm_{3.6e-05}$	$5.13e-05 \pm_{9.0e-07}$
AR-RRNS-10	$2.83e-04 \pm_{1.9e-04}$	$3.51e-04 \pm_{3.9e-05}$	$1.08e-04 \pm_{2.4e-05}$
AR-RRNS-11	$8.28e-04 \pm_{4.7e-04}$	$3.14e-04 \pm_{7.9e-06}$	$1.12e-04 \pm_{1.2e-05}$

En cuanto al número de soluciones obtenidas (ver **Tabla 9**) que pertenecen al Frente aproximado de Pareto, observamos que MOCell y NSGA-II tienen los mejores resultados. MOCell es 10.3% mejor que NSGA-II y 21.9% mejor que SPEA2. Por ejemplo, si nos centramos en el problema AR-RRNS-11, MOCell está en primer lugar, seguido de NSGA-II y SPEA2 (71, 68 y 53 soluciones óptimas de Pareto, respectivamente). Los datos en la Tabla 7 se representan como la media y la desviación estándar. El valor más alto es el mejor.

Tabla 9. Resultados del número de soluciones contenidas en el mejor frente encontrado. El mejor valor está marcado por un fondo sombreado.

Problema	SPEA2 $\overline{x}_{\sigma_n}$	NSGA-II $\overline{x}_{\sigma_n}$	MOCell $\overline{x}_{\sigma_n}$
AR-RRNS-8	$5.68e+01 \pm_{2.8e+0}$	$4.02e+01 \pm_{2.8e+0}$	$4.90e+01 \pm_{2.8e+0}$
AR-RRNS-9	$4.62e+01 \pm_{1.6e+0}$	$6.90e+01 \pm_{3.1e+0}$	$6.80e+01 \pm_{2.5e+0}$
AR-RRNS-10	$3.45e+01 \pm_{1.5e+0}$	$4.21e+01 \pm_{2.1e+0}$	$5.63e+01 \pm_{1.3e+0}$
AR-RRNS-11	$5.37e+01 \pm_{2.1e+0}$	$6.82e+01 \pm_{4.3e+0}$	$7.14e+01 \pm_{2.8e+0}$

Realizando 1000 iteraciones, los tres algoritmos genéticos encontraron excelentes aproximaciones al frente de Pareto en un tiempo aceptable: NSGA-II ($4.51e+02 \pm_{2.5e+01}$ ms), MOCell ($5.33e+02 \pm_{4.9e+01}$ ms) y SPEA2 ($5.34e+02 \pm_{2.4e+01}$ ms).

La **Figura 21** muestra el análisis bi-objetivo de la probabilidad de pérdida de información vs la redundancia, mientras que la **Figura 22** muestra el análisis bi-objetivo de la probabilidad de pérdida de información vs el tiempo de extracción. En ambos casos se puede observar que el algoritmo genético celular MOCell obtiene un frente que cubre una amplia gama de soluciones diferentes, así como también obtiene un mayor número de soluciones que integran el frente de Pareto aproximado.

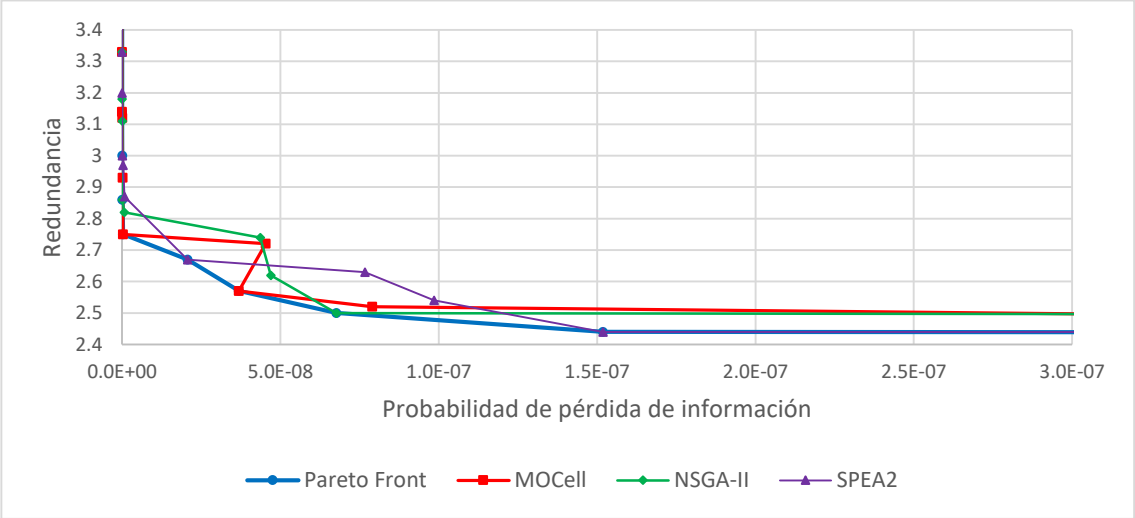

Figura 21. Ejemplos del frente de soluciones obtenido en el análisis bi-objetivo de: la probabilidad de pérdida de información vs la redundancia para la instancia del problema con 11 nubes: AR-RRNS-11.

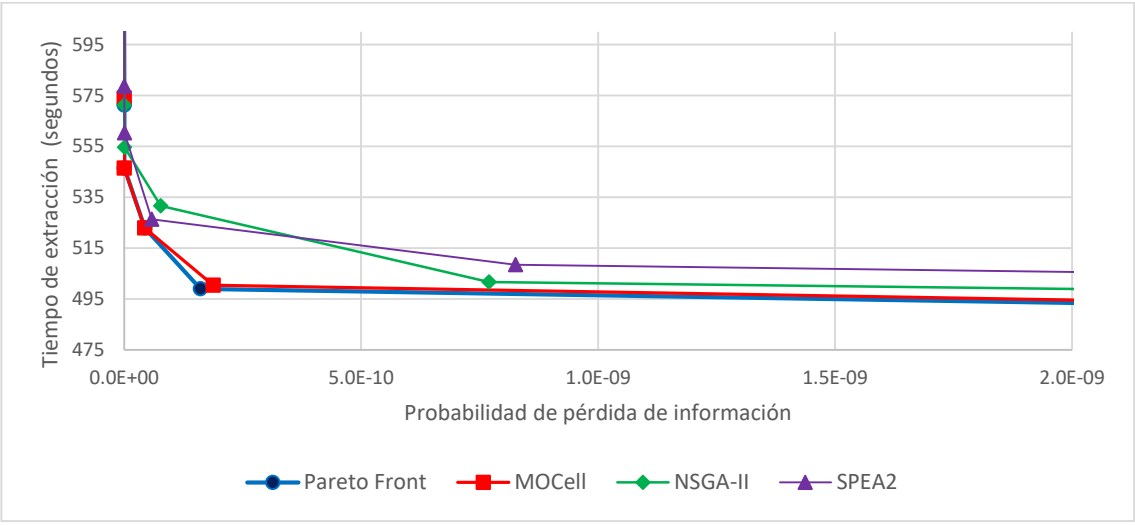

Figura 22. Ejemplos del frente de soluciones obtenido en el análisis bi-objetivo de: la probabilidad de pérdida de información vs el tiempo de extracción para la instancia del problema con 11 nubes: AR-RRNS-11.

5.4.1 Simulación de las variaciones en la velocidad de descarga y la probabilidad de falla de cada nube.

Se realizó un estudio experimental para simular las variaciones en la velocidad de descarga y la probabilidad de falla de cada nube. Nuevamente se compararon los algoritmos SPEA2, NSGA-II y MOCell. Se realizaron 60 simulaciones variando los valores de la velocidad de descarga y la probabilidad de falla, con una distribución uniforme entre el valor mínimo y el máximo, de las nubes. En cada simulación se utilizaron tres instancias del nuestro MOOP: VA-AR-RRNS-9 (9 nubes: $c_1, c_2, \dots, c_9$), VA-AR-RRNS-10 (10 nubes: $c_1, c_2, \dots, c_{10}$) y VA-AR-RRNS-11 (11 nubes: $c_1, c_2, \dots, c_{11}$). Se describen los resultados teniendo en cuenta los indicadores de calidad: EP, HV e IGD, el mejor valor está marcado por un fondo sombreado.

Considerando el indicador de calidad de Épsilon (EP) (ver **Tabla 10**), MOCell logra los mejores resultados (el valor más bajo con una desviación estándar más baja es mejor), seguido por NSGA-II y luego por SPEA2. Tiene un I_{EP} 63.1% mejor que NSGA-II y 82% mejor que SPEA2. En la **Figura 23** se muestra el diagrama de caja y bigote con los resultados para la instancia del problema VA-AR-RRNS-11.

La **Tabla 11** muestran los resultados al analizar el indicador de hipervolumen (HV), donde el valor más alto con una desviación estándar más baja es mejor. Los tres algoritmos alcanzaron resultados similares en este indicador, aunque MOCell fue ligeramente superior. La **Figura 24** presenta el diagrama de caja y bigote con los resultados del I_{HV} para la instancia del problema VA-AR-RRNS-11.

Tabla 10. Simulación de las variaciones en la velocidad de descarga y la probabilidad de falla de cada nube. Indicador de calidad EP. Media y desviación estándar, el mejor valor está marcado por un fondo sombreado.

Problema	SPEA2 $\overline{x}_{\sigma_n}$	NSGA-II $\overline{x}_{\sigma_n}$	MOCell $\overline{x}_{\sigma_n}$
VA-AR-RRNS-9	$7.66e-03 \pm_{9.1e-03}$	$2.16e-03 \pm_{1.3e-03}$	$1.03e-03 \pm_{7.7e-04}$
VA-AR-RRNS-10	$3.98e-03 \pm_{2.8e-03}$	$1.07e-03 \pm_{9.4e-05}$	$2.05e-05 \pm_{0.0e+00}$
VA-AR-RRNS-11	$3.67e-03 \pm_{1.2e-03}$	$4.26e-03 \pm_{7.1e-04}$	$1.71e-03 \pm_{2.0e-04}$

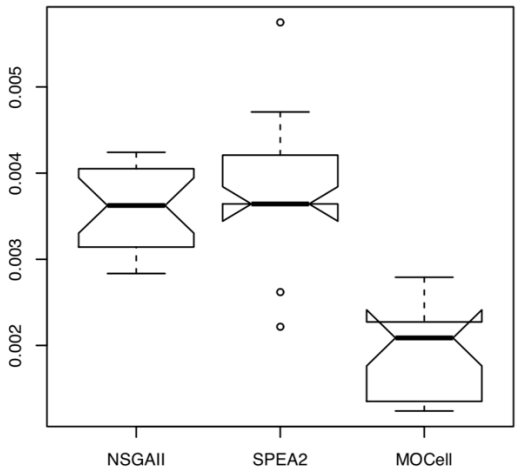

Figura 23. Simulación de las variaciones en la velocidad de descarga y la probabilidad de falla de cada nube. Indicador de calidad EP. Diagrama de caja y bigote para la instancia del problema VA-AR-RRNS-11. El valor más bajo con una desviación estándar más baja es mejor.

Tabla 11. Simulación de las variaciones en la velocidad de descarga y la probabilidad de falla de cada nube. Indicador de calidad HV. Media y desviación estándar, el mejor valor está marcado por un fondo sombreado.

Problema	SPEA2 $\overline{x}_{\sigma_n}$	NSGA-II $\overline{x}_{\sigma_n}$	MOCell $\overline{x}_{\sigma_n}$
VA-AR-RRNS-9	$9.77e-01 \pm_{2.1e-03}$	$9.78e-01 \pm_{7.7e-06}$	$9.78e{-}01 \pm_{2.0.e-05}$
VA-AR-RRNS-10	$9.83e-01 \pm_{3.0e-06}$	$9.83e-01 \pm_{4.4e-05}$	$9.83e-01 \pm_{1.0e-07}$
VA-AR-RRNS-11	$9.77e-01 \pm_{2.3e-05}$	$9.76e-01 \pm_{1.0e-03}$	$9.77e-01 \pm_{4.1e-06}$

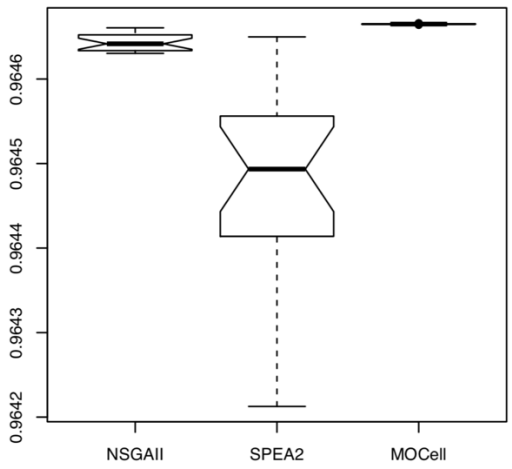

Figura 24. Simulación de las variaciones en la velocidad de descarga y la probabilidad de falla de cada nube. Indicador de calidad HV. Diagrama de caja y bigote para la instancia del problema VA-AR-RRNS-11. El valor más alto con una desviación estándar más baja es mejor.

Considerando el indicador IGD (ver **Tabla 12**), donde el valor más bajo con una desviación estándar más baja es mejor, MOCell supera al resto de los algoritmos, alcanzando un valor de I_{IGD} 55.9% mejor que SPEA2 y 63.2% mejor que NSGA-II. En la **Figura 25** se presenta el diagrama de caja y bigote con los resultados del I_{IGD} para la instancia del problema VA-AR-RRNS-11.

Tabla 12. Simulación de las variaciones en la velocidad de descarga y la probabilidad de falla de cada nube. Indicador de calidad IGD. Media y desviación estándar, el mejor valor está marcado por un fondo sombreado.

Problema	SPEA2 $\overline{x}_{\sigma_n}$	NSGA-II $\overline{x}_{\sigma_n}$	MOCell $\overline{x}_{\sigma_n}$
VA-AR-RRNS-9	$5.27e-04 \pm_{1.2e-04}$	$8.69e-04 \pm_{3.7e-04}$	$2.56e-04 \pm_{4.8e-05}$
VA-AR-RRNS-10	$5.50e-04 \pm_{3.2e-05}$	$2.13e-04 \pm_{1.8e-05}$	$1.55e-04 \pm_{3.7e-05}$
VA-AR-RRNS-11	$5.34e-04 \pm_{4.2e-05}$	$8.52e-04 \pm_{1.4e-04}$	$3.00e-04 \pm_{3.0e-05}$

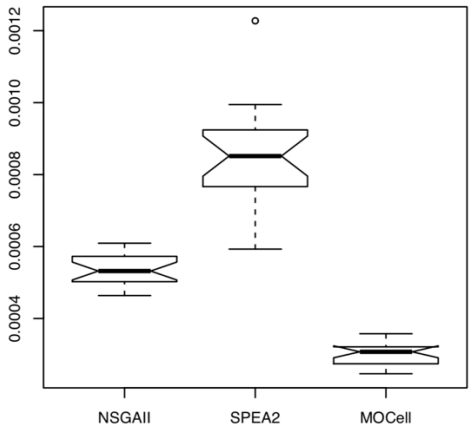

Figura 25. Simulación de las variaciones en la velocidad de descarga y la probabilidad de falla de cada nube. Indicador de calidad IGD. Diagrama de caja y bigote para la instancia del problema VA-AR-RRNS-11. El valor más bajo con una desviación estándar más baja es mejor.

5.4.2 Análisis de los resultados del frente de Pareto aproximado

La **Figura 26** nos permite comparar los valores de $P_r(k,n)$ y R para diferentes configuraciones (k,n), observándose que dichos objetivos de optimización están en conflicto. Por ejemplo, para la configuración $(2,11)$ se obtiene el menor valor de $P_r(k,n) = 2.37E-30$ que es el mejor valor para dicho objetivo, mientras que en ese mismo punto se obtiene mayor valor de $R = 11$ que es el peor valor. En el otro

extremo estaría, por ejemplo, la configuración $(10, 11)$ donde se obtiene $P_r(k, n) = 5.38E - 05$ (peor valor, por ser el mayor) versus $R = 2.2$ (mejor valor, por ser el menor).

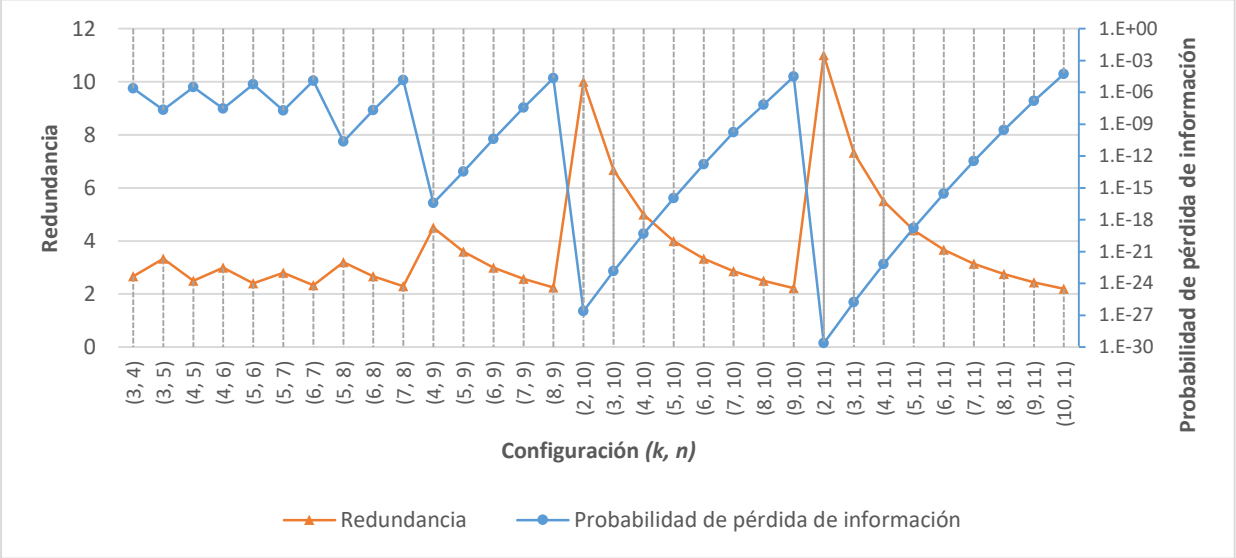

Figura 26. Comparación de valores de redundancia y probabilidad de pérdida de información para diferentes configuraciones (k, n) para la instancia del problema con 11 nubes: AR-RRNS-11.

De forma similar podemos observar, en la **Figura 27** los valores de $P_r(k, n)$ y T_{ex} para diferentes configuraciones (k, n), volviendo al ejemplo, para la configuración $(2, 11)$ se obtiene el menor valor de $P_r(k, n) = 2.37E - 30$ que es el mejor valor para dicho objetivo, contra el mayor valor de $T_{ex} = 2421.43$ segundos, que es el peor valor. Mientras que para la configuración $(10, 11)$ se obtiene $P_r(k, n) = 5.38E - 05$ (peor valor, por ser el mayor) versus $T_{ex} = 399.18$ segundos (mejor valor, por ser el menor).

La **Figura 28** muestra que los valores de redundancia (R) y tiempo de extracción (T_{ex}), no están en conflicto entre ellos, pues tienen comportamientos similares ante las diferentes configuraciones (k, n).

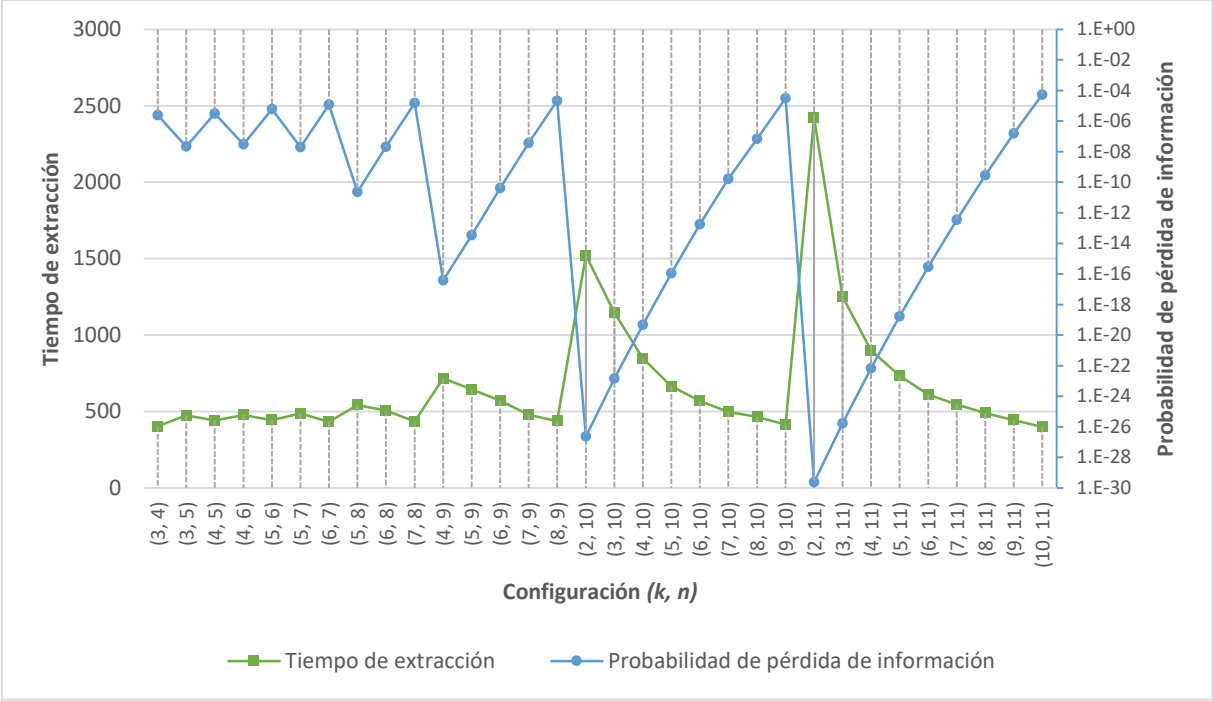

Figura 27. Comparación de valores de tiempo de extracción y probabilidad de pérdida de información para diferentes configuraciones (k, n) para la instancia del problema con 11 nubes: AR-RRNS-11.

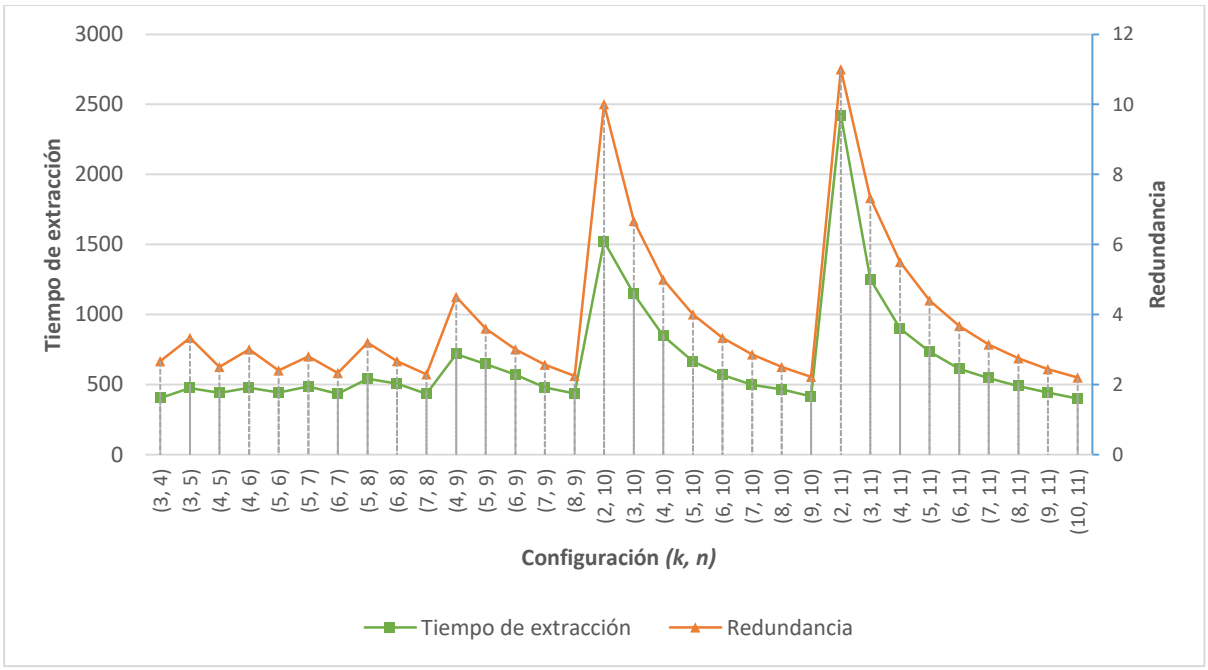

Figura 28. Comparación de valores de redundancia y tiempo de extracción para diferentes configuraciones (k, n) para la instancia del problema con 11 nubes: AR-RRNS-11.

La **Tabla 13** muestra ejemplos de soluciones obtenidas, indicando las nubes que deben usarse para almacenar la información, la configuración (k, n) del sistema AR-RRNS y los valores correspondientes de $P_r(k, n)$, R y T_{ex}. En el **Anexo 2** se muestran el resto de las soluciones obtenidas en una ejecución del algoritmo genético MOCell.

Tabla 13. Ejemplos de soluciones obtenidas por el algoritmo genético MOCell.

Nubes	(k, n)	$P_r(k, n)$	R	T_{ex}*(seg)*
$c_1, \dots, c_7, c_9, \dots, c_{11}$	*(2, 10)*	2.39E-27	10.00	1521.09
$c_1, \dots, c_7, , c_9, \dots, c_{11}$	*(8, 10)*	1.11E-07	2.50	452.73
$c_1, \dots, c_6, c_8, \dots, c_{11}$	*(9, 10)*	4.45E-05	2.22	406.31
$c_1, \dots, c_{11}$	*(2, 11)*	2.37E-30	11.00	2421.43
$c_1, \dots, c_{11}$	*(9, 11)*	1.52E-07	2.44	443.1
$c_1, \dots, c_{11}$	*(10, 11)*	5.38E-05	2.20	399.18

Los resultados muestran que las técnicas metaheurísticas son eficientes para resolver nuestro problema de optimización multi-objetivo AR-RRNS. Los algoritmos genéticos proporcionan al tomador de decisiones (a las capas superiores del sistema) un conjunto de configuraciones en lugar de solo una, lo que le permite elegir la solución más adecuada en diferentes situaciones.

El sistema de almacenamiento de datos seguro AR-RRNS en la nube múltiple incluirá en su proceso de configuración nuestro modelo de optimización multi-objetivo. Este sistema puede ser útil, por ejemplo, para garantizar la seguridad durante la transmisión y el procesamiento de datos médicos. Permitirá al usuario elegir el secreto (archivo) que se almacenará. El sistema seleccionará la configuración (k, n) y las nubes específicas para usar. Luego, se aplicarán algoritmos de intercambio de secretos y el secreto se dividirá en n fragmentos para ser almacenados en diferentes nubes. Cuando el usuario solicita acceder a sus datos, el sistema solo necesitará descargar fragmentos de k nubes que estén activas para recuperar la información original.

Capítulo 6. Conclusiones y trabajo futuro

En esta tesis se estudió el problema de la optimización multi-objetivo del sistema de almacenamiento en la nube AR-RRNS. Se propusieron mecanismos para determinar los parámetros configuración (k, n) para codificar/decodificar, junto con la selección de nubes específicas para almacenar la información. Se consideraron tres criterios de optimización: la probabilidad de pérdida de información, la redundancia y el tiempo de extracción. Se describió el modelo de almacenamiento de datos AR-RRNS con un esquema de umbral (k, n) y cómo este modelo se puede aplicar al entorno multi-nube. Se desarrolló un algoritmo, basado en algoritmos genéticos, para la optimización multi-objetivo de la configuración del modelo AR-RRNS, definiendo la codificación del cromosoma, la función de evaluación de aptitud, los parámetros y las restricciones adecuadas.

Se realizó un estudio experimental utilizando datos reales de 11 proveedores de almacenamiento en la nube. Comparamos el rendimiento de tres algoritmos genéticos: MOCell, NSGA-II y SPEA3 utilizando tres indicadores de calidad: distancia generacional invertida, épsilon aditivo e hipervolumen. Mostramos que MOCell supera a NSGA-II y SPEA2 en todos los indicadores de calidad. El I_{EP} de MOCell es 77.5% mejor que NSGA-II y 83.3% mejor que SPEA2. El número de soluciones pertenecientes a la aproximación al Frente de Pareto de MOCell es 10.3% mejor que NSGA-II y 21.9% mejor que SPEA2.

Demostramos que los algoritmos genéticos pueden ser utilizados de manera eficiente para la toma de decisiones en sistemas de almacenamiento en un entorno de multi-nube. Ya que encuentran, en un tiempo aceptable, un conjunto de soluciones aproximadas al Frente de Pareto. Por lo tanto, los GAs son una solución factible para realizar la configuración en línea del sistema.

Los resultados de la presente tesis fueron resumidos en el artículo: "*Multi-objective configuration of a secured distributed cloud data storage*", que fue presentado en la Conferencia de computación de alto rendimiento en América Latina (CARLA, *Latin America High Performance Computing Conference*) celebrada en Costa Rica del 25 al 27 de septiembre de 2019.

Como trabajo futuro, diseñaremos esquemas de compartición de secretos ponderados y estudiaremos varios mecanismos de inteligencia artificial y de inteligencia computacional para configurar el almacenamiento distribuido en la nube.

Literatura citada

Adham, A. M., Mohd-Ghazali, N., Ahmad, R. 2015. Performance optimization of a microchannel heat sink using the Improved Strength Pareto Evolutionary Algorithm (SPEA2). Journal of engineering thermophysics, 24(1), 86–100.

AlZain, M. A., Pardede, E., Soh, B., Thom, J. A. 2011. Cloud computing security: From single to multi-clouds. Proceedings of the Annual Hawaii International Conference on System Sciences, 5490–5499. doi:10.1109/HICSS.2012.153

Arora, R., Kaushik, S. C., Arora, R. 2015. Multi-objective and multi-parameter optimization of two-stage thermoelectric generator in electrically series and parallel configurations through NSGA-II. Energy, 91, 242–254.

Asmuth, C., Bloom, J. 1983. A modular approach to key safeguarding. IEEE Transactions on Information Theory, 29(2), 208–210. doi:10.1109/TIT.1983.1056651

Babenko, M., Chervyakov, N., Tchernykh, A., Kucherov, N., Shabalina, M., Vashchenko, I., Radchenko, G., Murga, D. 2017. Unfairness correction in P2P grids based on residue number system of a special form. En 2017 28th International Workshop on Database and Expert Systems Applications (DEXA), 2017, IEEE, pp. 147–151. IEEE. pp. 147–151.

Babitha M.P., Babu, K. R. R. 2016. Secure cloud storage using AES encryption. 2016 International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT), 859–864. doi:10.1109/ICACDOT.2016.7877709

Bader, J., Zitzler, E. 2011. HypE: An algorithm for fast hypervolume-based many-objective optimization. Evolutionary computation, 19(1), 45–76.

Barry, M., Schillinger, M., Weigt, H., Schumann, R. 2015. Configuration of hydro power plant mathematical models. En DA-CH Conference on Energy Informatics, 2015, Springer, pp. 200–207. Springer. pp. 200–207.

Basescu, C., Cachin, C., Haas, R., Antipolis, F.-S. 2011. Brief Announcement : Robust Data Sharing with Key-Value Stores. Distributed Computing, 3802, 221–222.

Bessani, A., Correia, M., Quaresma, B., Sousa, P., André, F., Sousa, P. 2013. DepSky. Proceedings of the sixth conference on Computer systems - EuroSys ’11, 00(00), 31. doi:10.1145/1966445.1966449

Beume, N., Naujoks, B., Emmerich, M. 2007. SMS-EMOA: Multiobjective selection based on dominated hypervolume. European Journal of Operational Research, 181(3), 1653–1669.

Blakley, G. R. 1979. Safeguarding cryptographic keys. Proceedings of the national computer conference, 48, 313. doi:10.1109/AFIPS.1979.98

Bleuler, S., Brack, M., Thiele, L., Zitzler, E. 2001. Multiobjective genetic programming: Reducing bloat using SPEA2. En Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546), 2001, IEEE, 1, pp. 536–543. IEEE. pp. 536–543.

Branke, J., Branke, J., Deb, K., Miettinen, K., Slowiński, R. 2008. Multiobjective optimization: Interactive and evolutionary approaches (Vol. 5252). Springer Science & Business Media.

Celesti, A., Fazio, M., Villari, M., Puliafito, A. 2016. Adding long-term availability, obfuscation, and encryption to multi-cloud storage systems. Journal of Network and Computer Applications, 59, 208–218.

Chervyakov, N., Babenko, M., Tchernykh, A., Kucherov, N. 2017. AR-RRNS : Configurable , Scalable and Reliable Systems for Internet of Things to Ensure Security. Future Generation Computer Systems, 1–15. doi:https://doi.org/10.1016/j.future.2017.09.061

Chervyakov, N., Babenko, M., Tchernykh, A., Kucherov, N., Miranda-López, V., Cortés-Mendoza, J. M. 2017. AR-RRNS: Configurable reliable distributed data storage systems for Internet of Things to ensure security. Future Generation Computer Systems. doi:10.1016/j.future.2017.09.061

Coello, C. 2017. List of References on Evolutionary Multiobjective Optimization. Consultado el 4 de julio de 2019, de CINVESTAV website: https://emoo.cs.cinvestav.mx/

Coello, C. A. C. 2018. Multi-objective optimization. Handbook of Heuristics, 1–28.

Coello, C. A. C. C., Pulido, G. T. 2001. A micro-genetic algorithm for multiobjective optimization. En International Conference on Evolutionary Multi-Criterion Optimization, 2001, Springer, pp. 126–140. Springer. pp. 126–140.

Coello, C., Lamont, G., Van Veldhuizen, D. 2007. Evolutionary algorithms for solving multi-objective problems (Vol. 5). Springer.

Corne, D. W., Jerram, N. R., Knowles, J. D., Oates, M. J. 2001. PESA-II: Region-based selection in evolutionary multiobjective optimization. En Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation, 2001, Morgan Kaufmann Publishers Inc., pp. 283–290. Morgan Kaufmann Publishers Inc. pp. 283–290.

Corne, D. W., Knowles, J. D., Oates, M. J. 2000. The Pareto envelope-based selection algorithm for multiobjective optimization. En International conference on parallel problem solving from nature, 2000, Springer, pp. 839–848. Springer. pp. 839–848.

Daemen, J., Rijmen, V. 2003. The Rijndael Block Cipher: AES Proposal. En Nist.

Day, R. O., Lamont, G. B. 2005. An effective explicit building block MOEA, the MOMGA-IIa. En 2005 IEEE Congress on Evolutionary Computation, 2005, IEEE, 1, pp. 17–24. IEEE. pp. 17–24.

Deb, K. 2014. Multi-objective optimization. En Search methodologies. Springer. pp. 403–449.

Deb, K., Jain, H. 2013. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. IEEE Transactions on Evolutionary Computation, 18(4), 577–601.

Deb, K., Jain, P., Gupta, N. K., Maji, H. K. 2004. Multiobjective placement of electronic components using evolutionary algorithms. IEEE transactions on components and packaging technologies, 27(3), 480–492.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6(2), 182–197. doi:10.1109/4235.996017

Deb, K., Sindhya, K., Hakanen, J. 2016. Multi-Objective Optimization. En Decision Sciences (Vol. 24). doi:10.1201/9781315183176-4

Deb, K., Steuer, R. E., Tewari, R., Tewari, R. 2011. Bi-objective portfolio optimization using a customized hybrid NSGA-II procedure. En International Conference on Evolutionary Multi-Criterion Optimization, 2011, Springer, pp. 358–373. Springer. pp. 358–373.

Durillo, J. J., Nebro, A. J. 2011. jMetal: A Java framework for multi-objective optimization. Advances in Engineering Software, 42(10), 760–771.

Durillo, J. J., Zhang, Y., Alba, E., Nebro, A. J. 2009. A study of the multi-objective next release problem. En 1st International Symposium on Search Based Software Engineering, 2009, IEEE, pp. 49–58. IEEE. pp. 49–58.

Eiben, A. E. 2003. Multiparent Recombination in Evolutionary Computing. doi:10.1007/978-3-642-18965-4_6

Eiben, A. E., Smith, J. E. 2015. Introduction to Evolutionary Computing. doi:10.1007/978-3-662-44874-8

Erickson, M., Mayer, A., Horn, J. 2001. The niched pareto genetic algorithm 2 applied to the design of groundwater remediation systems. En International Conference on Evolutionary Multi-Criterion Optimization, 2001, Springer, pp. 681–695. Springer. pp. 681–695.

Ermakova, T., Fabian, B. 2013. Secret sharing for health data in multi-provider clouds. En Business Informatics (CBI), 2013 IEEE 15th Conference on, 2013, IEEE, pp. 93–100. IEEE. pp. 93–100.

Fonseca, C. M., Fleming, P. J. 1993. Genetic Algorithms for Multiobjective Optimization: FormulationDiscussion and Generalization. En Icga, 1993, Citeseer, 93(July), pp. 416–423. Citeseer. pp. 416–423.

Fourman, M. P. 1985. Compaction of symbolic layout using genetic algorithms. En Genetic Algorithms and Their Applications: Proc. 1st Int. Conf. Genetic Algorithms, Princeton, Lawrence Erlbaum, NJ, 1985, 1985.

Frincu, M. E., Craciun, C. 2011. Multi-objective meta-heuristics for scheduling applications with high availability requirements and cost constraints in multi-cloud environments. En Utility and Cloud Computing (UCC), 2011 Fourth IEEE International Conference on, 2011, IEEE, pp. 267–274. IEEE. pp. 267–274.

García, S., Quintana, D., Galván, I. M., Isasi, P. 2011. Portfolio optimization using SPEA2 with resampling. En International Conference on Intelligent Data Engineering and Automated Learning, 2011, Springer, pp. 127–134. Springer. pp. 127–134.

Gen, M., Lin, L. 2007. Genetic algorithms. Wiley Encyclopedia of Computer Science and Engineering, 1–15.

Goldberg, D. E. 1989. Genetic algorithms in search, optimization, and machine learning. Addison-Wesley Longman Publishing Co., Inc.

Goldberg, D. E., Deb, K. 1991. A comparative analysis of selection schemes used in genetic algorithms. En Foundations of genetic algorithms (Vol. 1). Elsevier. pp. 69–93.

Goyal, K. K., Jain, P. K., Jain, M. 2012. Optimal configuration selection for reconfigurable manufacturing system using NSGA II and TOPSIS. International Journal of Production Research, 50(15), 4175–4191.

Guzek, M., Pecero, J. E., Dorronsoro, B., Bouvry, P. 2014. Multi-objective evolutionary algorithms for energy-aware scheduling on distributed computing systems. Applied Soft Computing, 24, 432–446.

Hajela, P., Lin, C.-Y. 1992. Genetic search strategies in multicriterion optimal design. Structural optimization, 4(2), 99–107.

Hochberg, J., TAMHANE, A. C. 1987. Multiple comparison procedures. John Wiley & Sons,.

Holland, J. H. (John H. 1992. Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence. MIT Press.

Jemai, J., Zekri, M., Mellouli, K. 2012. An NSGA-II algorithm for the green vehicle routing problem. En European Conference on Evolutionary Computation in Combinatorial Optimization, 2012, Springer, pp. 37–48. Springer. pp. 37–48.

Khishtandar, S., Zandieh, M. 2017. Comparisons of some improving strategies on NSGA-II for multi-objective inventory system. Journal of Industrial and Production Engineering, 34(1), 61–69.

Kita, H., Yabumoto, Y., Mori, N., Nishikawa, Y. 1996. Multi-objective optimization by means of the thermodynamical genetic algorithm. En International Conference on Parallel Problem Solving from Nature, 1996, Springer, pp. 504–512. Springer. pp. 504–512.

Knowles, J. D., Corne, D. W. 2000. Approximating the nondominated front using the Pareto archived evolution strategy. Evolutionary computation, 8(2), 149–172.

Konak, A., Coit, D. W., Smith, A. E. 2006. Multi-objective optimization using genetic algorithms: A tutorial. Reliability Engineering & System Safety, 91(9), 992–1007.

Kursawe, F. 1990. A variant of evolution strategies for vector optimization. En International Conference on Parallel Problem Solving from Nature, 1990, Springer, pp. 193–197. Springer. pp. 193–197.

Laumanns, M., Ocenasek, J. 2002. Bayesian optimization algorithms for multi-objective optimization. En International Conference on Parallel Problem Solving from Nature, 2002, Springer, pp. 298–307. Springer. pp. 298–307.

Liefooghe, A., Derbel, B. 2016. A correlation analysis of set quality indicator values in multiobjective optimization. En Proceedings of the Genetic and Evolutionary Computation Conference 2016, 2016, ACM, pp. 581–588. ACM. pp. 581–588.

Lin, S.-J., Chung, W.-H., Han, Y. S. 2014. Novel Polynomial Basis and Its Application to Reed-Solomon Erasure Codes. En Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science, Washington, DC, USA, 2014, IEEE Computer Society, pp. 316–325. doi:10.1109/FOCS.2014.41

Lopez-Falcon, E., Tchernykh, A., Chervyakov, N., Babenko, M., Nepretimova, E., Miranda-López, V., Drozdov, A. Y., Radchenko, G., Avetisyan, A. 2018. Adaptive encrypted cloud storage model. En Young Researchers in Electrical and Electronic Engineering (EIConRus), 2018 IEEE Conference of Russian, 2018, IEEE, pp. 329–334. IEEE. pp. 329–334.

Lu, H., Yue, T., Ali, S., Zhang, L. 2016. Nonconformity resolving recommendations for product line configuration. En 2016 IEEE International Conference on Software Testing, Verification and Validation (ICST), 2016, IEEE, pp. 57–68. IEEE. pp. 57–68.

Maestrini, P. 1973. Redundant Residue Number. C(3), 307–315.

Marium, S., Nazir, Q., Ahmed Shaikh, A., Ahthasham, S., Aamir Mehmood, M. 2012. Implementation of Eap with RSA for Enhancing The Security of Cloud Computing. International Journal of Basic and Applied Sciences, 1.

Marler, R. T., Arora, J. S. 2004. Survey of multi-objective optimization methods for engineering. Structural and multidisciplinary optimization, 26(6), 369–395.

Mell, P. M., Grance, T. 2011. The NIST definition of cloud computing. doi:10.6028/NIST.SP.800-145

Mignotte, M. 1983. How To Share a Secret. Communications of the ACM (CACM), 22(1), 612–613. doi:http://doi.acm.org/10.1145/359168.359176

Miranda-López, V., Tchernykh, A., Cortés-Mendoza, J. M., Babenko, M., Radchenko, G., Nesmachnow, S., Du, Z. 2018. Experimental Analysis of Secret Sharing Schemes for Cloud Storage Based on RNS. En E. Mocskos & S. Nesmachnow (Eds.), High Performance Computing. Springer International Publishing: Cham. pp. 370–383.

Naujoks, B., Willmes, L., Bäck, T., Haase, W. 2002. Evaluating multi-criteria evolutionary algorithms for airfoil optimisation. En International Conference on Parallel Problem Solving from Nature, 2002, Springer, pp. 841–850. Springer. pp. 841–850.

Nebro, A. J., Durillo, J. J., Luna, F., Dorronsoro, B., Alba, E. 2009. MOCell: A cellular genetic algorithm for multiobjective optimization. International Journal of Intelligent Systems, 24(7), 726–746.

Nebro, A. J., Durillo, J. J., Vergne, M. 2015. Redesigning the jMetal multi-objective optimization framework. En Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation, 2015, ACM, pp. 1093–1100. ACM. pp. 1093–1100.

Okabe, T., Yaochu Jin, Sendhoff, B. 2003. A critical survey of performance indices for multi-objective optimisation. En The 2003 Congress on Evolutionary Computation, 2003. CEC '03., 2003, IEEE, 2, pp. 878–885. doi:10.1109/CEC.2003.1299759

Osyczka, A., Kundu, S. 1995. A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. Structural optimization, 10(2), 94–99.

Paar, C., Pelzl, J. 2010. Understanding Cryptography. doi:10.1007/978-3-642-04101-3

Pulido, G. T., Coello, C. A. C. 2003. The micro genetic algorithm 2: Towards online adaptation in evolutionary multiobjective optimization. En International Conference on Evolutionary Multi-Criterion Optimization, 2003, Springer, pp. 252–266. Springer. pp. 252–266.

Pundkar, S. N., Shekokar, N. 2016. Cloud computing security in multi-clouds using Shamir's secret sharing scheme. International Conference on Electrical, Electronics, and Optimization Techniques, ICEEOT 2016, 392–395. doi:10.1109/ICEEOT.2016.7755427

Rabin, M. O. 1989. Efficient dispersal of information for security, load balancing, and fault tolerance. Journal of the ACM, 36(2), 335–348. doi:10.1145/62044.62050

Rao, S. 1984. Multiobjective optimization in structural design with uncertain parameters and stochastic processes. AIAA Journal, 22(11), 1670–1678. doi:10.2514/3.8834

Rao, S. S. 2009. Engineering optimization : theory and practice (4a ed.). John Wiley & Sons, Inc.

Rathanam, G. J., Sumalatha, M. R. 2014. Dynamic secure storage system in cloud services. 2014 International Conference on Recent Trends in Information Technology, ICRTIT 2014. doi:10.1109/ICRTIT.2014.6996175

Reed, I. S., Solomon, G. 1960. Polynomial codes over certain finite fields. Journal of the Society for Industrial and Applied Mathematics, 300–304. doi:https://doi.org/10.1137/0108018

rey Horn, J., Nafpliotis, N., Goldberg, D. E. 1994. A niched Pareto genetic algorithm for multiobjective optimization. En Proceedings of the first IEEE conference on evolutionary computation, IEEE world congress on computational intelligence, 1994, Citeseer, 1, pp. 82–87. Citeseer. pp. 82–87.

Riquelme, N., Von Lücken, C., Baran, B. 2015. Performance metrics in multi-objective optimization. En 2015 Latin American Computing Conference (CLEI), 2015, IEEE, pp. 1–11. IEEE. pp. 1–11.

Rivest, R. L., Shamir, A., Adleman, L. 1978. A method for obtaining digital signatures and public-key cryptosystems. Communications of the ACM, 21(2), 120–126. doi:10.1145/359340.359342

Schaffer, D. 1984. Some experiments in machine learning using vector evaluated genetic algorithms. Nashville, Tennessee.

Shamir, A. 1979. How to share a secret. Communications of the ACM, 22(11), 612–613. doi:10.1145/359168.359176

Sofianopoulos, S., Tambouratzis, G. 2011. Studying the spea2 algorithm for optimising a pattern-recognition based machine translation system. En 2011 IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making (MDCM), 2011, IEEE, pp. 97–104. IEEE. pp. 97–104.

Soyel, H., Tekguc, U., Demirel, H. 2011. Application of NSGA-II to feature selection for facial expression recognition. Computers & Electrical Engineering, 37(6), 1232–1240.

Srinivas, N., Deb, K. 1994. Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evolutionary computation, 2(3), 221–248.

Tchernykh, A., Babenko, M., Chervyakov, N., Miranda-López, V., Kuchukov, V., Cortés-Mendoza, J. M., Deryabin, M., Kucherov, N., Radchenko, G., Avetisyan, A. 2018. AC-RRNS: Anti-collusion secured data sharing scheme for cloud storage. International Journal of Approximate Reasoning, 102, 60–73. doi:10.1016/j.ijar.2018.07.010

Tchernykh, A., Babenko, M., Miranda-López, V., Drozdov, A. Y., Avetisyan, A. 2018. WA-RRNS: Reliable Data Storage System Based on Multi-cloud. En 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2018, IEEE, pp. 666–673. IEEE. pp. 666–673.

Tos, U., Mokadem, R., Hameurlain, A., Ayav, T., Bora, S. 2015. Dynamic replication strategies in data grid systems: a survey. Journal of Supercomputing, 71(11), 4116–4140. doi:10.1007/s11227-015-1508-7

Van Veldhuizen, D. A. 1999. Multiobjective evolutionary algorithms: classifications, analyses, and new innovations. AIR FORCE INST OF TECH WRIGHT-PATTERSONAFB OH SCHOOL OF ENGINEERING.

Van Veldhuizen, D. A., Lamont, G. B. 1998. Multiobjective evolutionary algorithm research: A history and analysis. Citeseer.

Wolpert, D. H., Macready, W. G. 1997. No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation, 1(1), 67–82. doi:10.1109/4235.585893

Wu, J., Ping, L., Ge, X., Wang, Y., Fu, J. 2010. Cloud storage as the infrastructure of cloud computing. En Intelligent Computing and Cognitive Informatics (ICICCI), 2010 International Conference on, 2010, IEEE, pp. 380–383. IEEE. pp. 380–383.

Zhang, Q., Li, H. 2007. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on evolutionary computation, 11(6), 712–731.

Zhang, W., Hansen, K. M. 2009. An evaluation of the NSGA-II and MOCell genetic algorithms for self-management planning in a pervasive service middleware. En 2009 14th IEEE International Conference on Engineering of Complex Computer Systems, 2009, IEEE, pp. 192–201. IEEE. pp. 192–201.

Zhu, Z., Zhang, G., Li, M., Liu, X. 2015. Evolutionary multi-objective workflow scheduling in cloud. IEEE Transactions on parallel and distributed Systems, 27(5), 1344–1357.

Zitzler, E., Thiele, L. 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. IEEE Transactions on Evolutionary Computation, 3(4), 257–271. doi:10.1109/4235.797969

Zitzler, Eckart, Deb, K., Thiele, L. 2000. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation, 8(2), 173–195. doi:10.1162/106365600568202

Zitzler, Eckart, Laumanns, M., Thiele, L. 2001. SPEA2: Improving the strength Pareto evolutionary algorithm. TIK-report, 103.

Zitzler, Eckart, Thiele, L., Laumanns, M., Fonseca, C. M., Da Fonseca Grunert, V. 2002. Performance assessment of multiobjective optimizers: An analysis and review. TIK-Report, 139.

Zydallis, J. B., Van Veldhuizen, D. A., Lamont, G. B. 2001. A statistical comparison of multiobjective evolutionary algorithms including the MOMGA-II. En International Conference on Evolutionary Multi-Criterion Optimization, 2001, Springer, pp. 226–240. Springer. pp. 226–240.

Anexos

Anexo 1. Resultados experimentales del modelo de almacenamiento de datos AR-RRNS. Configuración (k, n) vs Redudancia, tiempo de codificación y tiempo de decodificación.

Configuración $(\boldsymbol{k}, \boldsymbol{n})$	Redundancia	Tiempo de codificación (segundos)	Tiempo de decodificación (segundos)
(2, 4)	4.00	37.63	354.96
(3, 4)	2.67	25.66	296.18
(4, 4)	2.00	20.09	269.50
(2, 5)	5.00	40.08	472.91
(3, 5)	3.33	30.31	354.24
(4, 5)	2.50	22.87	326.80
(5, 5)	2.00	22.53	292.93
(2, 6)	6.00	51.08	567.10
(3, 6)	4.00	35.49	433.62
(4, 6)	3.00	26.39	368.28
(5, 6)	2.40	26.00	333.78
(6, 6)	2.00	23.35	293.20
(2, 7)	7.00	57.92	636.15
(3, 7)	4.67	41.12	485.77
(4, 7)	3.50	31.41	412.74
(5, 7)	2.80	30.24	374.42
(6, 7)	2.33	27.97	337.15
(7, 7)	2.00	24.81	306.36
(2, 8)	8.00	65.47	756.47
(3, 8)	5.33	46.25	552.41
(4, 8)	4.00	32.97	455.53
(5, 8)	3.20	31.79	394.46
(6, 8)	2.67	29.13	369.37
(7, 8)	2.29	26.60	323.95
(8, 8)	2.00	23.61	303.01
(2, 9)	9.00	74.77	806.42
(3, 9)	6.00	51.24	605.95
(4, 9)	4.50	39.05	493.77
(5, 9)	3.60	36.22	441.51
(6, 9)	3.00	34.07	409.11
(7, 9)	2.57	29.79	350.38
(8, 9)	2.25	26.29	328.97
(9, 9)	2.00	25.06	301.11
(2, 10)	10.00	84.97	838.49

(3, 10)	6.67	55.24	614.82
(4, 10)	5.00	41.59	520.69
(5, 10)	4.00	39.79	436.35
(6, 10)	3.33	35.69	396.58
(7, 10)	2.86	31.65	358.53
(8, 10)	2.50	28.73	345.72
(9, 10)	2.22	26.66	314.36
(10, 10)	2.00	26.11	295.72
(2, 11)	11.00	87.12	914.58
(3, 11)	7.33	59.66	646.07
(4, 11)	5.50	44.02	529.79
(5, 11)	4.40	42.09	473.61
(6, 11)	3.67	38.60	412.24
(7, 11)	3.14	34.03	386.46
(8, 11)	2.75	30.80	356.12
(9, 11)	2.44	28.87	329.70
(10, 11)	2.20	27.69	301.23
(11, 11)	2.00	25.61	292.26

Anexo 2. Resultados obtenidos con el algoritmo MOCell para el MOOP AR-RRNS con 11 nubes.

Nubes utilizadas	**Configuración** $(\boldsymbol{k}, \boldsymbol{n})$	**Probabilidad de pérdida de información**	**Redundancia**	**Tiempo de extracción** (segundos)
c_2, c_5, c_9, c_{10}	*(3, 4)*	2.40E-06	2.67	404.73
c_3, c_9, c_{10}, c_{11}	*(4, 4)*	3.03E-03	2.00	340.74
c_1, c_3, c_4, c_8, c_9	*(3, 5)*	2.17E-08	3.33	475.16
$c_1, c_6, c_9, c_{10}, c_{11}$	*(4, 5)*	3.13E-06	2.50	441.35
$c_1, c_3, c_4, c_8, c_9, c_{11}$	*(4, 6)*	2.95E-08	3.00	477.57
$c_1, c_2, c_6, c_9, \dots, c_{11}$	*(5, 6)*	5.98E-06	2.40	443.77
$c_1, c_2, c_5, c_8, \dots, c_{11}$	*(5, 7)*	1.93E-08	2.80	486.78
$c_1, c_3, c_6, c_8, \dots, c_{11}$	*(6, 7)*	1.20E-05	2.33	432.71
$c_1, \dots, c_3, c_6, c_8, \dots, c_{11}$	*(5, 8)*	2.34E-11	3.20	541.11
$c_1, c_2, c_6, \dots, c_{11}$	*(6, 8)*	2.07E-08	2.67	507.06
$c_1, c_2, c_6, \dots, c_{11}$	*(7, 8)*	1.48E-05	2.29	434.44
$c_1, \dots, c_4, c_6, c_8, \dots, c_{11}$	*(4, 9)*	3.89E-17	4.50	717.92
$c_1, \dots, c_3, c_6, \dots, c_{11}$	*(5, 9)*	3.45E-14	3.60	645.36
$c_1, \dots, c_2, c_5, \dots, c_{11}$	*(6, 9)*	4.02E-11	3.00	570.84
$c_1, \dots, c_2, c_5, \dots, c_{11}$	*(7, 9)*	3.69E-08	2.57	479.69
$c_1, \dots, c_2, c_5, \dots, c_{11}$	*(8, 9)*	2.12E-05	2.25	436.24
$c_1, \dots, c_6, c_8, \dots, c_{11}$	*(2, 10)*	2.39E-27	10.00	1521.09
$c_1, c_3, \dots, c_{11}$	*(3, 10)*	1.44E-23	6.67	1146.74
$c_1, \dots, c_4, c_6, \dots, c_{11}$	*(4, 10)*	4.81E-20	5.00	847.48
$c_1, \dots, c_4, c_6, \dots, c_{11}$	*(5, 10)*	1.09E-16	4.00	665.19
$c_1, c_3, \dots, c_{11}$	*(6, 10)*	1.72E-13	3.33	569.44
$c_1, c_3, \dots, c_{11}$	*(7, 10)*	1.69E-10	2.86	497.80
$c_1, \dots, c_3, c_5, \dots, c_{11}$	*(8, 10)*	6.77E-08	2.50	464.91
$c_1, \dots, c_3, c_5, \dots, c_{11}$	*(9, 10)*	3.13E-05	2.22	415.17
$c_1, \dots, c_{11}$	*(2, 11)*	2.37E-30	11.00	2421.43
$c_1, \dots, c_{11}$	*(3, 11)*	1.66E-26	7.33	1250.46
$c_1, \dots, c_{11}$	*(4, 11)*	6.60E-23	5.50	900.16
$c_1, \dots, c_{11}$	*(5, 11)*	1.68E-19	4.40	734.89
$c_1, \dots, c_{11}$	*(6, 11)*	2.88E-16	3.67	611.51
$c_1, \dots, c_{11}$	*(7, 11)*	3.39E-13	3.14	546.46
$c_1, \dots, c_{11}$	*(8, 11)*	2.76E-10	2.75	489.13
$c_1, \dots, c_{11}$	*(9, 11)*	1.52E-07	2.44	443.10
$c_1, \dots, c_{11}$	*(10, 11)*	5.38E-05	2.20	399.18