UNIVERSIDAD DE LA REPÚBLICA URUGUAY

FACULTAD DE INGENIERÍA

CECAL

Instituto de Computación, Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay

PROYECTO DE GRADO INGENIERÍA EN COMPUTACIÓN

Diseño de redes de contenido en plataformas cloud

Gerardo Goñi
gerardo.goni@fing.edu.uy

Octubre de 2024

Supervisor:
Santiago Iturriaga, Universidad de la República.

Co-Supervisor:
Sergio Nesmachnow, Universidad de la República.

Diseño de redes de contenido en plataformas cloud

Resumen

En este trabajo se estudia el escenario donde existe un agente externo que puede ser un broker virtual o un proveedor de servicios, el cual actúa como intermediario entre el comprador de un servicio de cloud y los vendedores de ese servicio. El negocio del agente externo consiste en optimizar la asignación de recursos reservados y la calidad de servicio para poder construir redes de distribución de contenido para distintos proveedores, en función de las demandas de sus clientes. El problema es modelado considerando la ubicación geográfica de cada centro de datos y cada cliente que solicita un recurso a una CDN, de modo de poder estimar la calidad de servicio que se ofrece a los usuarios. Se utiliza un algoritmo evolutivo multiobjetivo para resolver el problema de optimización de manera de minimizar el costo para el agente externo y maximizar la calidad de servicio a los usuarios. El análisis experimental es realizado sobre instancias reales construidas con herramientas especializadas. Los resultados del algoritmo evolutivo son comparados con heurísticas propuestas en trabajos relacionados previos. El algoritmo evolutivo desarrollado logra mejorar en el mejor caso un 90,39 % a la mejor de las heurísticas implementadas basados en trabajos previos.

Índice general

Capítulo 1

Introducción

Las redes de distribución de contenido (Content Delivery Network, CDN por su sigla en inglés) son redes distribuidas que entregan servicios y contenido a los usuarios, constituyen infraestructuras clave para proporcionar servicios de Internet escalables globalmente y garantizar acuerdos de nivel de servicios específicos entre proveedores de servicios y usuarios finales, lo que permite ofrecer una alta calidad de servicio (QoS). Las CDNs distribuyen diversos tipos de contenido y servicios, como transmisión de video, descargas de software, contenido web/móvil, almacenamiento en caché, evaluación de rendimiento, equilibrio de carga, análisis e inteligencia en la nube. El objetivo principal de una CDN es distribuir contenido con alta disponibilidad y rendimiento. Para ello, las CDNs deben contar con numerosos servidores distribuidos en centros de datos globales. Este modelo es económicamente inviable para pequeños proveedores de contenido sin grandes infraestructuras informáticas, quienes optan por alquilar servicios de proveedores de CDNs grandes como Akamai, Google Cloud CDN, y Microsoft Azure CDN.

En los últimos años ha habido una tendencia a utilizar la elasticidad y distribución global de los servicios en la nube para construir CDNs basadas en la nube (Gao et al., 2015; Hu et al., 2014; Jokhio et al., 2013; Xiao et al., 2016). Este enfoque implica alquilar infraestructura bajo el modelo de Infraestructura como Servicio (IaaS) y distribuir el contenido globalmente para ofrecer una QoS competitiva. Sin embargo, un desafío significativo de este enfoque es el aprovisionamiento de recursos en la nube, un problema difícil conocido y generalizado en soluciones de software basadas en la nube (Zhang et al., 2016).

Este proyecto de grado presenta una estrategia para resolver el problema de aprovisionamiento de recursos para CDNs basadas en la nube, optimizando tanto las métricas del sistema como las de usuario. Se propone un modelo de optimización multiobjetivo que considera los costos de IaaS, incluyendo alquiler de máquinas virtuales (VM), almacenamiento y ancho de banda, y la QoS para los usuarios finales. El modelo incluye características de plataformas en la nube, como ubicación geográfica de recursos, descuentos por compras a granel y alquiler de instancias reservadas. Además, se introduce un modelo comercial general con un agente virtual (Nesmachnow et al., 2015), que aplica un enfoque Multi-Tenancy para reducir costos al gestionar múltiples proveedores de contenido simultáneamente, aprovechando precios con descuento para grandes volúmenes de máquinas virtuales y estrategias de intercambio de recursos.

Para comparar los resultados del AE se implementaron tres heurísticas y ser resolvió el problema de forma exacta para algunas instancias del problema. Las heurísticas

programadas fueron: Greedy en costo, Greedy en QoS y Round Robin. Las soluciones de forma exacta del problema se obtuvieron con software especializado que se describe en la Sección 5.5. Lo que caracteriza a las heurísticas implementadas es que asignan a un centro de datos cada pedido de recurso de contenido siguiendo una política que: selecciona el centro de datos de mejor costo (Greedy en costo), selecciona el centro de datos de mejor QoS (Greedy en QoS) y en el caso de Round Robin se asignan los recursos siguiendo un orden secuencial al momento de seleccionar los centros de datos. En el caso de la solución de forma exacta se desarrolló un modelo matemático que minimiza el costo sujeto a una restricción de cumplimiento de un mínimo de QoS dado por un umbral. El AE propuesto en este proyecto de grado tiene tres fases: inicialización de la población, planificación y enrutamiento. Las dos primeras fases determinan la asignación de máquinas virtuales a los centros de datos. La fase de enrutamiento determina cuál es la máquina virtual que atiende cada pedido de contenido, por lo tanto también determina la asignación de recursos de contenidos en los centros de datos.

Para el caso de instancias que permitieron calcular las soluciones con el método exacto las soluciones del AE estuvieron muy cercanas, en diversidad y precisión, a las soluciones exactas. En el caso en el que el tamaño de las instancias del problema solo permiten encontrar soluciones con heurísticas, el AE mostró mejoras significativas respecto a las heurísticas de referencias utilizadas.

El resto del informe se organiza de la siguiente manera. El Capítulo 2 describe el problema de construcción de redes de distribución de contenido en plataformas cloud. En el Capítulo 3 se presenta una reseña de trabajos previos vinculados con la temática de este proyecto de grado. El Capítulo 4 describe la metodología utilizada y los resultados obtenidos en el proceso de medición del RTT entre regiones geográficas, también en dicho capítulo se comparan las mediciones de RTT con las distancias geográficas entre regiones. En el Capítulo 5 se describen las heurísticas base de comparación y el algoritmo evolutivo implementado para resolver el problema planteado. El Capítulo 6 presenta los resultados obtenidos al evaluar el algoritmo evolutivo propuesto para resolver el problema. Por último, en el Capítulo 7 se desarrollan las conclusiones del trabajo y los principales líneas de trabajo futuro propuestas.

Capítulo 2

Problema de diseño de redes de contenido en plataformas cloud

Este capítulo brinda una descripción del problema de diseño de redes de contenido en plataformas cloud. En la Sección 2.1 introduce conceptos generales referentes al problema, la Sección 2.2 describe el problema y la Sección 2.3 presentan el modelo y la formulación matemática del problema.

2.1. Conceptos generales

En esta sección se describen los conceptos generales vinculados con el problema de diseño de redes de distribución de contenido en una plataforma cloud.

2.1.1. Redes de distribución de contenido en plataformas cloud

En la actualidad el modelo de computación en las plataformas cloud (Buyya et al., 2010; Foster et al., 2008) es uno de los modelos informáticos más utilizados. Mediante este modelo se puede acceder bajo demanda a recursos compartidos (capacidad de almacenamiento, servidores, redes, aplicaciones y servicios) que son asignados y liberados de forma dinámica con una mínima gestión por parte de los proveedores de servicios cloud.

Los usuarios de las plataformas cloud perciben que los recursos son ilimitados y están siempre disponibles dado que los recursos se asignan y liberan, la mayoría de las veces, de manera automática. Si una plataforma cloud no logra cubrir la demanda de recursos, puede obtener recursos adicionales de otras plataformas cloud al recurrir a la técnica de *cloud bursting* (Mattess et al., 2017) y así cubrir la demanda de los usuarios.

Los proveedores de computación en plataformas cloud presentan una pila de tres modelos de servicio que caracterizan sus productos. Cada modelo se diferencia en función de las capas en las que el usuario tiene control, las capas van desde el *hardware* hasta el *software*. Los modelos de servicio que caracterizan los productos de los proveedores son:

- Infraestructura como servicio (IaaS), En este modelo el hardware se encuentra virtualizado en la plataforma cloud lo que permite ajustar los recursos asignados según las necesidades computacionales. Los proveedores de este servicio tienen centros de datos distribuidos por todo el mundo y brindan recursos tales como almacenamiento, máquinas virtuales, infraestructura de redes, etc.

- Plataforma como servicio (PaaS), En cuanto al nivel de abstracción este modelo se encuentra por arriba del modelo de IaaS. En este modelo los proveedores de servicios cloud proporcionan un entorno en donde los desarrolladores de software pueden construir aplicaciones escalables y con alta disponibilidad sin preocuparse de las actualizaciones de software, la infraestructura de red, el sistem operativo o el almacenamiento. En el modelo PaaS la plataforma cloud gestiona de manera automática la cantidad de recursos asignados a la aplicación desarrollada por el usuario.
- Software como servicio (SaaS), En cuanto al nivel de abstracción este modelo se encuentra por arriba del modelo de PaaS y representa la capa más alta de la pila. Generalmente este modelo se considera un servicio orientado a usuarios finales. El proveedor de servicio cloud instala, administra y mantiene el software el cual se puede acceder a través de la web o las interfaces de programación de aplicaciones que brinda el proveedor de servicio cloud.

Una CDN es un sistema distribuido de servidores alojados en múltiples centros de datos en Internet. Las CDNs sirven una gran parte del contenido de Internet (música, documentos, vídeos, imágenes, recursos web, etc.) al mismo tiempo que ofrecen los contenidos a usuarios finales con alta disponibilidad y alto rendimiento. La alta disponibilidad y el alto rendimiento se logran gracias a un mejor balanceo de la carga en los servidores que alojan los contenidos y en los enlaces que interconectan las distintas subredes de Internet.

La distribución geográfica de los servidores que componen a una CDN permite servir los datos a los usuarios en función de la cercanía geográfica a cada servidor. Las CDNs proporcionan protección contra ataques de denegación de servicio (DoS) mediante el uso de su infraestructura de servidores distribuidos para absorber el tráfico de ataque (Marinescu, 2017).

Es financieramente prohibitivo para los proveedores de contenido pequeños competir a gran escala con los proveedores de servicio de CDNs convencionales mediante la implementación de nuevos centros de datos. En una CDN en la plataforma cloud (CDNC) se puede ajustar dinámicamente los arrendamientos de ancho de banda, máquinas virtuales y recursos de almacenamiento en función de la demanda de contenido para reducir el costo total de arrendamiento sin sacrificar la QoS brindada a los usuarios finales. La CDNC debe permitir que usuarios finales de la red accedan a contendidos compartidos por un conjunto de proveedores de contenido y a su vez la CDNC debe garantizar que no exista interferencia entre los distintos proveedores de contenido.

El modelo de negocio de IaaS en las plataformas cloud crea nuevas oportunidades para que los proveedores de contenido pequeños puedan usar CDNs rentables y escalables sin inversiones en la instalación y el mantenimiento de la infraestructura, por lo tanto hay una tendencia creciente en la investigación para aprovechar las ventajas que brinda desarrollar CDNs en las plataformas cloud (Gao et al., 2015; Hu et al., 2014; Jokhio et al., 2013; Xiao et al., 2016). Junto con la tendencia creciente en el uso de las plataformas cloud para el diseño de CDNs se introdujo el problema de aprovisionamiento de recursos en estas plataformas el cual es un problema difícil de abordar (Zhang et al., 2016).

2.1.2. Broker virtual

Con la finalidad de gestionar las distintas ofertas de los proveedores de servicios en las plataformas cloud surgió un agente denominado *cloud broker* el cual oficia de intermediario entre los usuarios y los proveedores de servicios cloud (Chhetri et al., 2013).

Generalmente el servicio ofrecido por el *cloud broker* consiste en asistir a los usuarios de las plataformas cloud para que encuentren a los proveedores que mejor se ajustan a sus requerimientos y la mejor manera de implementar sus aplicaciones, de acuerdo con los requisitos específicos de rendimiento, acuerdos de nivel de servicio, seguridad y costos.

En este proyecto de grado se utiliza un tipo especial de *cloud broker* denominado *broker virtual* el cual fue introducido por Nesmachnow et al. (2015). Los recursos virtuales ofrecidos por el *cloud broker* consisten en máquinas virtuales que son rentadas a distintos proveedores de servicios cloud por un largo período de tiempo con un descuento significativo en el precio. El descuento obtenido por el *cloud broker* le permite rentar a los usuarios finales instancias de máquinas virtuales bajo demanda a un menor precio que los proveedores de servicios cloud. El *broker virtual* que se presenta en este proyecto de grado tiene como objetivo minimizar el costo total de la CDNC construida en los centros de datos de los proveedores de IaaS a la vez que tiene en cuenta la QoS ofrecida a los usuarios finales que consumen los servicios.

Existen dos razones por las que una CDNC utilizada por múltiples proveedores de contenido a la vez suele ser más barata que una CDNC utilizada por un solo proveedor. La primera razón es que los proveedores de servicio IaaS generalmente otorgan descuentos por volumen al arrendar recursos en una misma infraestructura, de esta manera una CDNC que cuenta con múltiples proveedores de contenido se beneficia de la suma de demanda de recursos de los proveedores. La segunda razón es que en una CDNC con múltiples proveedores de contenido se pueden aprovechar más instancias de máquinas virtuales reservadas dado que cada máquina virtual reservada puede ser usada por distintos proveedores de contenido según las demandas de sus usuarios. Rentar de manera anticipada instancias de máquinas virtuales por un periodo de tiempo prolongado es más barato que rentar máquinas virtuales a demanda.

La Figura 2.1 muestra un esquema de cómo interaccionan el *broker virtual*, los proveedores de contendido, los proveedores de servicio IaaS y los usuarios finales. Los proveedores de contenido contratan el servicio de CDNC que ofrece el *broker virtual*, el *broker virtual* contrata tiempo de uso y espacio de almacenamiento en instancias de máquinas virtuales a los proveedores de servicio IaaS y los usuarios finales utilizan la CDNC construida por el *broker virtual* para consumir los recursos de los proveedores de contenidos. El *broker virtual* se encarga de la gestión y asignación de recursos a cada proveedor de contenido de acuerdo a la demanda de sus usuarios. Los proveedores de servicio de IaaS otorgan descuentos al *broker virtual* por volumen de recursos rentados y a su vez el *broker virtual* aprovecha mejor el tiempo de uso de las instancias de máquinas virtuales reservadas al poder usarlas con múltiples proveedores de contenido.

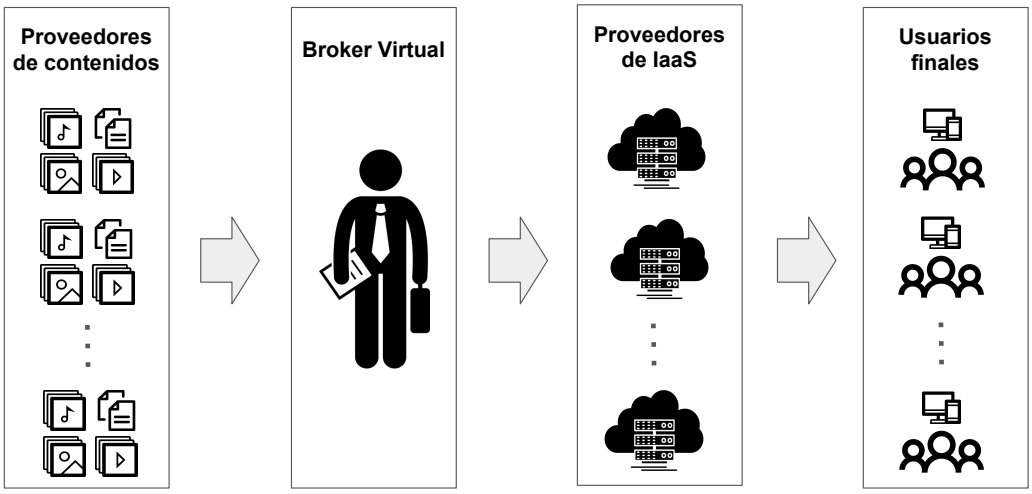

Figura 2.1: Diagrama que representa la interacción entre los proveedores de contendido, el broker virtual, los proveedores de servicio IaaS y los usuarios finales.

2.2. Descripción del problema

El problema de optimización planteado en este proyecto de grado consiste en encontrar una asignación de recursos que permita al *broker virtual* maximizar sus ingresos y simultáneamente maximizar la QoS brindada a los usuarios. El *broker virtual* basado en la demanda de recursos de los usuarios en cada instante de tiempo, tiene en cuenta el costo y la QoS al momento de asignar un recurso en una máquina virtual determinada en un centro de datos dado. Para cada solicitud de contenido, junto a la información de que zona geográfica proviene, se deberá determinar que tipo de instancia de máquina virtual (reservada o a demanda) y desde que centro de datos se atenderá esa solicitud.

Como se muestra en el esquema de la la Figura 2.2, cada centro de datos puede atender solicitudes de contenidos que provienen desde distintas regiones geográficas. Los proveedores de contenidos utilizan la CDNC construida por el *broker virtual* para compartir sus recursos con usuarios distribuidos en distintas zonas geográficas. El *broker virtual* renta instancias de máquina virtual a los proveedores de IaaS para alojar la CDNC en centros de datos que están distribuidos geográficamente. Por lo tanto es necesario que el *broker virtual* tenga en cuenta la QoS que existe entre las regiones geográficas en donde están los distintos centros de datos y cada región en donde se encuentran los usuarios que consumen los recursos de la CDNC construida.

En el Capítulo 4 se presenta la métrica de QoS utilizada en este proyecto de grado, la cual se mide entre las regiones geográficas donde están los usuarios que consumen los recursos y las regiones geográficas donde están los centros de datos. De manera adicional a la QoS, el *broker virtual* debe tener en cuenta el costo de la CDNC que consiste en la suma del costo de almacenamiento de los recursos en cada centro de datos, el costo de la transferencia de datos y el costo de rentar tiempo de uso de instancias de máquinas virtuales (reservadas y a demanda) en los centros de datos.

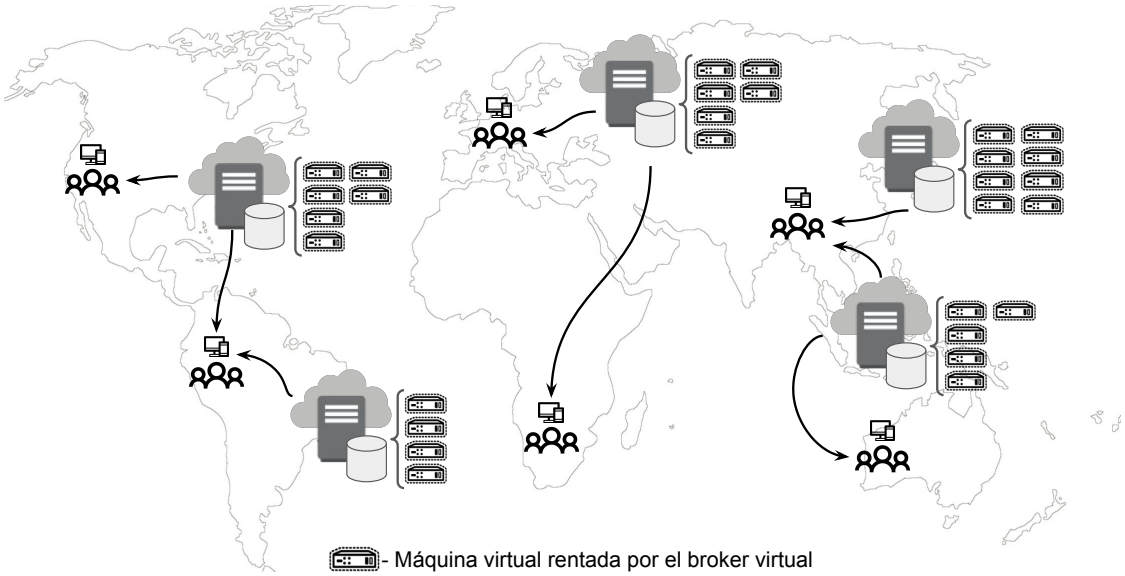

Figura 2.2: La red de distribución de contenidos en la plataforma cloud construida por el *broker virtual* se aloja en instancias de máquinas virtuales en centros de datos distribuidos geográficamente, cada dentro de datos atiende solicitudes de recursos de distintas zonas geográficas.

2.3. Modelo del problema y formulación matemática

Esta sección introduce el modelo propuesto para una CDNC y la formulación matemática del problema de optimización multiobjetivo que minimiza el costo relacionado con el diseño de CDNC y maximiza la calidad de servicio brindado por la CDNC a los usuarios.

2.3.1. Modelo del problema

El modelo del problema de diseño de CDNC considera la asignación estática de recursos en la plataforma cloud y un conjunto de rutas dinámicas que los usuarios utilizan para acceder al contenido, durante un período de tiempo $[0, T]$. La asignación estática de recursos en la plataforma cloud permanece fija durante todo el período de tiempo $[0, T]$. Sin embargo, cada acceso a los recursos es dinamicamente enrutado dependiendo del tiempo $t \in [0, T]$ en el que se esté accediendo al contenido. Los elementos considerados en la formulación del problema son:

- Un conjunto de proveedores de contenido $P = \{p_1, ..., p_f\}$.
- Un conjunto de contenidos $K = \{k_1, ..., k_n\}$ que los proveedores comparten con sus usuarios.
- Un conjunto de regiones geográficas $R = \{r_1, ..., r_s\}$ a las que pertenecen los usuarios. Cada usuario pertenece a una única región.
- Un conjunto de instancias de máquinas virtuales reservadas que se rentan durante todo el período de tiempo $[0, T]$.
- Un conjunto de instancias de máquinas virtuales que se rentan a demanda $\{mvd_1, ..., mvd_d\}$ durante un período de tiempo $[t_{iv}, t_{jv}] \subseteq [0, T], v = 1, 2, ..., d$.
- Un conjunto de centros de datos $C = \{c_1, ..., c_m\}$ en donde se rentan las instancias

de máquinas virtuales. Cada centro de datos c_e tiene un costo de transferencia DTC (data transfer cost), un costo por almacenamiento DSC (data storage cost) y un costo de arrendamiento de cómputo CRC (computation renting cost). El valor de CRC_e para un centro de datos c_e se desglosa en costo de arrendamiento de instancia de máquina virtual bajo demanda y costo de arrendamiento de instancia de máquina virtual reservada. La función $DTC_e(d)$ define el costo económico de transferir un volumen de datos d desde el centro de datos c_e a Internet. La función $DSC_e(d)$ define el costo económico de almacenar un volumen de datos d en el centro de datos c_e.

- Por simplicidad, se asume que los bloques de contenido tienen el mismo tamaño (en bytes) KS.
- Cada máquina virtual tiene la capacidad de procesar hasta CR accesos a contenidos de forma simultanea.
- Se deben comenzar a transmitir en el instante de tiempo t todas las solicitudes de recursos k_i hechas en el instante de tiempo t.
- Las instancias de máquinas virtuales bajo demanda son rentadas por hora, independientemente de si se usa o no la hora completa. Una vez rentada la instancia de una máquina virtual, se cobra por toda la hora.

Las variables en el modelo propuesto son:

- Se define un conjunto variables reales Q_{le} que indica la métrica de QoS que el centro de datos c_e provee a los usuarios de la región r_l.
- Se define la variable binaria kp_{ij} que indica si el contenido k_i es compartido por el proveedor p_j.
- Se define la variable rk_{li}^t que indica el número de accesos al contenido k_i por los usuarios en la región r_l en un tiempo t.
- Se define la función $D(t)$ que dado un instante de tiempo $t \in T$, retorna el conjunto de recursos k_i demandados en el instante t.
- Se define la función VD_e como el costo económico de rentar una instancia de máquina virtual bajo demanda en el centro de datos c_e durante un instante de tiempo t. La función VR_e define el costo económico de rentar una instancia de máquina virtual reservada en el centro de datos c_e durante todo el tiempo $[0, T]$.
- Un conjunto de variables binarias x_{ie}, donde $x_{ie} = 1$ indica que el contenido k_i está almacenado en el centro de datos c_e.
- Un conjunto de variables enteras y_e^t, donde y_e^t indica la cantidad de instancias de máquinas virtuales asignadas en el centro de datos c_e en el instante de tiempo t.
- Un conjunto de variables enteras $\tilde{y}_e$, donde $\tilde{y}_e$ indica la cantidad de instancias de máquina virtuales reservadas a arrendar en el centro de datos c_e durante todo el período de tiempo $[0, T]$.
- Un conjunto de variables enteras $z_{lie}^{vt} = b$, donde $z_{lie}^{vt} = b$ indica que los usuarios de la región rl descargan un total b de veces el contenido k_i desde la máquina virtual v del centro de datos c_e en el instante de tiempo t.
- Se define la función $M(c_e, t)$ que dado un centro de datos c_e y un instante de tiempo t, retorna el conjunto de máquinas virtuales rentadas en el centro de datos c_e en el instante de tiempo t.

2.3.2. Formulación matemática

La formulación matemática del problema está dada por la Ecuación 2.1. El modelo del problema tiene dos funciones objetivos representadas por la Ecuación 2.1a y la Ecuación 2.1b y un conjunto de restricciones representado por las Ecuaciones 2.1i, 2.1j, 2.1k y 2.1l.

$$min \quad f^{cost} = \sum_{c_e \in C} f^{mv}(c_e) + \sum_{c_e \in C} f^{store}(c_e) + \sum_{c_e \in C} f^{net}(c_e) \tag{2.1a}$$

$$min \quad f^{qos} = \sum_{r_l \in R} \sum_{c_e \in C} \left(f^{qos\_rc}(r_l, c_e) \right) \tag{2.1b}$$

$$con \tag{2.1c}$$

$$f^{mv}(c_e) = \tilde{y}_e \times VR_e + \sum_{t \in \{0..T\}} VD_e \times \max\{0, y_e^t - \tilde{y}_e\} \tag{2.1d}$$

$$f^{store}(c_e) = DSC_e \left(\sum_{k_i \in K} x_{ie} \times KS \right) \tag{2.1e}$$

$$f^{net}(c_e) = DTC_e \left(\sum_{t \in \{0..T\}} \sum_{k_i \in K} \sum_{r_l \in R} \sum_{v \in M(c_e,t)} z_{lie}^{vt} \times KS \right) \tag{2.1f}$$

$$f^{qos\_rc}(r_l, c_e) = Q_{le} \times \sum_{t \in \{0..T\}} \sum_{k_i \in K} \sum_{v \in M(c_e,t)} z_{lie}^{vt} \tag{2.1g}$$

sujeto a (2.1h)

$$rk_{li}^t = \sum_{c_e \in C} \sum_{v \in M(c_e,t)} z_{lie}^{vt}, \forall t \in \{0..T\}, k_i \in K, r_l \in R \tag{2.1i}$$

$$z_{lie}^{vt} > 0,\ z_{l'i'e}^{vt} > 0 \Longrightarrow kp_{ij} = 1,\ kp_{i'q} = 1,\ j = q$$
$$\forall t \in \{0..T\}, k_i \in K, r_l \in R, c_e \in C, v \in M(c_e, t), p_j \in P, p_q \in P \tag{2.1j}$$

$$z_{lie}^{vt} \leq CR, \forall t \in \{0..T\}, k_i \in K, r_l \in R, c_e \in C, v \in M(c_e, t) \tag{2.1k}$$

$$z_{lie}^{vt} \times x_{ie} \geqslant z_{lie}^{vt}, \forall t \in \{0..T\}, k_i \in K, r_l \in R, c_e \in C, v \in M(c_e, t) \tag{2.1l}$$

El objetivo del problema de optimización de la construcción de una red de distribución de contenido en la plataforma cloud es calcular una solución que minimiza simultáneamente el costo total de infraestructura dado por f^{cost} (Ecuación 2.1a) y la métrica de QoS dada por f^{qos} (Ecuación 2.1b).

El costo total de la infraestructura f^{cost} se define como la suma del costo de rentar las máquinas virtuales, el costo total de almacenamiento de datos, y el costo total de la transferencia de datos. La Ecuación 2.1d representa al costo de rentar máquinas virtuales en el centro de datos c_e el cual se define como la suma del costo económico de rentar instancias de máquinas virtuales bajo demanda en el centro de datos c_e durante un instante de tiempo t (VD_e) y el costo económico de rentar instancias de máquinas virtuales reservadas en el centro de datos c_e durante todo el tiempo T (VR_e).

El costo de almacenamiento en el centro de datos c_e resulta de aplicar la función DSC_e a la suma del tamaño de todos los recursos almacenados en el centro de datos c_e (Ecuación 2.1e).

El costo de transferencia desde el centro de datos c_e a Internet resulta de aplicar la función DSC_e a la suma del tamaño de todos los recursos transferidos desde el centro de datos c_e hacia Internet (Ecuación 2.1f).

La métrica de QoS total calculada por la función f^{qos} representada en la (Ecuación 2.1b) se define como la suma de las métricas de QoS que hay entre cada región de usuario r_l y cada centro de datos c_e durante todo el tiempo de planificación. La función $f^{qos\_rc}(r_l, c_e)$ representada en la (Ecuación 2.1g) calcula, para todo el tiempo de planificación, la QoS entre r_l y c_e. En la función $f^{qos\_rc}(r_l, c_e))$ se cuentan cuantos recursos de contenidos de cada tipo se transfieren entre r_l y c_e, ese valor se multiplica por la métrica de calidad de servicios entre r_l y c_e (Q_{le}).

Se deben satisfacer las siguientes restricciones:

a) Debe ser satisfecha la demanda de cada recurso k_i en cada instante de tiempo t y cada región r_l (Ecuación 2.1i).

b) Una instancia de máquina virtual no puede brindar contenidos de dos proveedores distintos a la vez. (Ecuación 2.1j).

c) Cada máquina virtual podrá procesar hasta CR peticiones simultaneas (Ecuación 2.1k).

d) Un centro de datos c_e puede atender pedidos de un recurso k_i solo si k_i está almacenado en c_e (Ecuación 2.1l).

Capítulo 3

Trabajos relacionados

Este capítulo presenta una revisión de trabajos relacionados con el aprovisionamiento de recursos en plataformas en la nube, lo cual está relacionado con el problema de diseño y optimización de CDNs basados en la nube.

3.1. Introducción

Variedad de trabajos en la literatura relacionada han reconocido que para tener un modelo de negocio rentable es fundamental contar con políticas efectivas de aprovisionamiento de recursos y algoritmos precisos. Este enfoque general no solo se aplica a la provisión de servicios en Internet, sino también a varios otros campos como la gestión de la cadena de suministro, telecomunicaciones, etc. (Park and Willinger, 2000), (Crandall et al., 2009).

La literatura muestra que los algoritmos efectivos de aprovisionamiento de recursos han sido ampliamente estudiados y son fundamentales en el caso particular de la provisión de servicios en la nube (Zhang et al., 2016).

3.2. Aprovisionamiento de recursos en la nube

Diferentes características y criterios de optimización han sido considerados al modelar el problema de aprovisionamiento de recursos en la nube. En lo que resta de la sección se presentan algunos trabajos relacionados.

Con la finalidad de permitir la transmisión de video a través de dispositivos heterogéneos y redes variables, Gao et al. (2015) propusieron la optimización del costo de la transcodificación de video en un sistema de transmisión de velocidad de bits adaptativa. Los autores plantearon codificar cada contenido en diferentes velocidades de bits y dividirlos en segmentos. Algunos de los segmentos generan costos de almacenamiento al almacenarse en caché; otros generan costos de computación dado que se transcodifican en línea. El objetivo es minimizar el costo general a largo plazo determinando si un segmento debe almacenarse en caché o transcodificarse en línea. Los autores utilizaron la optimización de Lyapunov para minimizar el costo de adquisición de infraestructura de computación y almacenamiento para la transcodificación de video en línea. Los escenarios usados para evaluar el método propuesto fueron construidos utilizando estadísticas de la plataforma de videos YouTube (patrones de visualización, popularidad y tasas de llegada

de solicitudes). También para la construcción de los escenarios se usó información de costos de IaaS (computación y almacenamiento) de Amazon S3 y Amazon EC2 en instancias bajo demanda. Los experimentos demostraron que el método propuesto puedo reducir el 30 % del costo operativo, en comparación con el esquema de almacenar en caché todos los segmentos. El modelo del problema planteado no considera instancias reservadas de máquinas virtuales ni una métrica para evaluar la QoS al usuario final (solo se optimiza la utilización de recursos en el proveedor de servicios).

Para reducir el costo total de IaaS ofrecido por un proveedor de servicios en la nube Jokhio et al. (2013) también se centraron solo en el proveedor de servicios y plantearon un esquema donde se intercambia almacenamiento por computación, o viceversa. Se propuso un algoritmo en línea que plantea una estrategia de compensación de computación y almacenamiento para una transcodificación de video rentable en la nube. La estrategia propuesta implementa un sistema de estimación del costo computacional, el costo de almacenamiento y la popularidad individual de cada video transcodificado. Los datos mencionados son utilizados para tomar decisiones estratégicas sobre el tiempo de almacenamiento y la frecuencia con la que se debe volver a transcodificar a partir de un video fuente específico. El enfoque propuesto no considera QoS ni instancias de VM reservadas. La evaluación experimental utilizó patrones de carga semi-sintéticos, datos reales de acceso a videos de la plataforma Bambuser y costos de IaaS de Amazon Web Services. Los resultados obtenidos demostraron que el algoritmo propuesto basado en puntuaciones presentó una mejor rentabilidad en comparación con dos estrategias base intuitivas. De manera adicional, el algoritmo propuesto logró un buen equilibrio entre la utilización de recursos computacionales y de almacenamiento para patrones de carga tanto semi-sintéticos como realistas.

Xiao et al. (2016) plantearon minimizar el costo de alquiler de VMs para ejecutar aplicaciones de video con uso intensivo de computación en una CDN de video basada en la nube. Las VMs se alquilan en múltiples centros de datos distribuidos geográficamente que estén cerca de los solicitantes de video.Se formuló el problema de redirigir dinámicamente las solicitudes y aprovisionar recursos, mientras que se equilibra el ahorro en costo y la QoS, como un problema de optimización estocástica. Se diseñó un algoritmo en línea basado en el marco de optimización de Lyapunov para resolver el problema planteado. La evaluación experimental se realizó en conjuntos de datos sintéticos construidos con trazas de YouTube y otros servicios de videos, además de un conjunto de datos aleatorios generados con la utilización de la distribución de Poisson. El algoritmo propuesto minimizó el costo promedio a largo plazo de alquilar recursos en la nube mientras se mantiene la QoS de los usuarios.El trabajo tuvo en cuenta la ubicación geográfica de los centros de datos y los descuentos por alquiler masivo de máquinas virtuales. El trabajo no tuvo en cuenta las máquinas virtuales con instancias reservadas ni costos de tráfico de red y almacenamiento.

Hu et al. (2014) minimizaron los costos de almacenamiento y de alquiler de las máquinas virtuales bajo demanda mediante la optimización del aprovisionamiento de recursos y la colocación de réplicas para CDNs basadas en la nube con énfasis en el manejo de patrones de demanda dinámicos. La optimización la hicieron asegurando un mínimo de QoS para los usuarios finales. Se propuso un algoritmo en línea de dos niveles, que incluye heurísticas greedy e iterativas para resolver tanto el problema de aprovisionamiento y almacenamiento en caché a largo plazo como el problema de almacenamiento en caché y equilibrio de solicitudes a corto plazo. El algoritmo propuesto consta de 2 pasos. El

primer paso maximiza las demandas totales atendidas por los recursos cuyo tiempo de contratación no está vencido. El segundo paso minimiza el costo total del alquiler de nuevos recursos para satisfacer todas las demandas restantes. Finalmente los autores proponen el algoritmo de almacenamiento en caché y equilibrio de solicitudes para ajustar de manera dinámica (en tiempo de ejecución) la ubicación de contenidos y el ruteo. Este último algoritmo es liviano y se puede ejecutar con frecuencia sobre las soluciones del primer algoritmo para maximizar las demandas totales. Los métodos propuestos, en el marco de evaluaciones con escenarios sintéticos, exhibieron un rendimiento superior en comparación con algoritmos de planificación y réplicas de renombre.El trabajo propuesto por los autores solo consideró el alquiler de máquinas virtuales bajo demanda, pero no máquinas virtuales reservadas.Usaron la distancia geográfica como estimador de delay y simularon el RTT mediante una fórmula que depende de la distancia.

Liu et al. (2021) formularon un algoritmo en línea para resolver el problema de aprovisionamiento de recursos en la nube donde maximizan la QoS respetando un costo preestablecido. En el trabajo presentado definen que la QoS experimentada por una empresa en un período de tiempo t es función de la cantidad de recursos de la nube que puede contratar en el período t. Se considera que el costo de rentar máquinas virtuales en la nube varía para cada período de tiempo t. El objetivo del problema propuesto por los autores es maximizar la QoS acumulada en el tiempo, sujeto al presupuesto de costos totales para el alquiler de recursos . En el trabajo realizaron simulaciones basadas en trazas del mundo real. Los resultados de los algoritmos propuestos superaron significativamente las líneas de base en una amplia gama de entornos. En este trabajo no se consideran costos de transferencia, almacenamiento ni de máquinas virtuales reservadas. La QoS que se utiliza en este trabajo es desde el punto de vista de la empresa que presta el servicio y no del usuario.

El análisis de investigaciones previas reveló que diversos enfoques han sido propuestos para abordar diferentes variantes del problema planteado en este trabajo. Sin embargo, pocos estudios han focalizado en la optimización simultánea de métricas asociadas al proveedor de servicios en la nube (costo) y a los usuarios finales (QoS). En este trabajo, se propone extender los enfoques generales de estudios anteriores, incluyendo la optimización simultánea de los costos del alquiler de máquinas virtuales, almacenamiento y ancho de banda de red, así como QoS ofrecida al usuario final. En los trabajos previos estudiados, cuando la QoS es estimada se hace sobre datos sintéticos, en este trabajo la QoS se estima en base a mediciones reales de RTT entre las regiones geográficas consideradas. También se incorpora la ubicación geográfica, descuentos por volumen e instancias reservadas a largo plazo. En este trabajo se integra al Broker Virtual el cual oficia de intermediario con un enfoque multi-tenant. El Broker Virtual contribuye a reducir los costos mediante el aprovechamiento de descuentos por volumen y el uso compartido de recursos.

Capítulo 4

Estimación de QoS basada en el round-trip time

En este capítulo se describe el proceso de medición del RTT (round-trip time, por su sigla en inglés) entre zonas geográficas. Se presenta el criterio de selección de las regiones geográficas y diseño de los escenarios de medición. Se explica la metodología utilizada en el análisis de los datos de las mediciones de RTT. Por último, se describen los resultados obtenidos y se compara la distancia entre regiones geográficas y el resultado de las mediciones del RTT.

4.1. Medición del RTT entre zonas geográficas

En esta sección se describe la metodología utilizada en al proceso de medición del RTT entre pares de regiones geográficas. Se consideró la información de tráfico de Internet y las zonas de disponibilidad de recursos de los proveedores de servicios en la nube para modelar dos conjuntos de regiones geográficas: RCD - regiones que albergan centros de datos, y $RUSR$ -regiones donde se ubican los usuarios que utilizan el servicio. Para medir el RTT entre cada par de regiones $\langle RCD, RUSR \rangle$, se emplearon mediciones del tipo PING de la red RIPE Atlas, según lo descrito por Jacobsen (2015).

4.1.1. Red RIPE atlas

RIPE Atlas es una plataforma de medición de parámetros de red en Internet, pone a disposición de sus miembros recursos que permiten realizar mediciones de parámetros de red. El organismo encargado de llevar adelante este proyecto es RIPE NCC, que cuenta con diversos colaboradores en todo el mundo. La plataforma se encuentra en una etapa de expansión, enfocándose principalmente en Asia, Latinoamérica y Caribe, y África.

RIPE Atlas se basa en sensores de medida desplegados sobre Internet. La red está formada de dos tipos de sensores:

- Probes: medidores de fácil instalación con los cuales se pueden generar mediciones utilizando muy poco ancho de banda. Los probes tienen la característica de no monitorer la red en la cual se los conecta.

- Anchors: medidores con los cuales se puede generar un número mayor de mediciones que con los probes. Los anchors son el objetivo de muchas mediciones iniciadas por otros dispositivos, por lo tanto las redes que los contengan deben tener alto ancho de banda y alta disponibilidad.

La red RIPE Atlas permite realizar experimentos definidos por el usuario o mediciones definidas por el usuario, denominadas UDMs. Cada medición tiene un costo, por lo tanto el usuario que quiera hacer mediciones debe tener un saldo favorable de créditos. Las formas de obtener créditos son: a) teniendo medidores propios y dejándolos conectados a Internet, para que los demás medidores RIPE Atlas puedan usarlo para las mediciones y b) mediante transferencias de créditos entre cuentas de usuarios de la red RIPE Atlas.

Las mediciones que se pueden hacer con los anchors y los probes son: ping, traceroute, DNS, HTTP GET, SSLcert. En este proyecto de grado solo se utilizaron mediciones del tipo ping para determinar el RTT entre cada par de regiones $\langle RCD, RUSR \rangle$.

4.1.2. Selección de las regiones geográficas y diseño de los escenarios de medición

La Figura 4.1 muestra la distribución geográficas de las regiones utilizadas. Las regiones *RCD* fueron seleccionadas en las zonas en las que la empresa Amazon tiene centros de datos que brindan el servicio Amazon EC2 descrito por Antunes (2016). Las regiones *RUSR* se seleccionaron según su cantidad de población.

Figura 4.1: Distribución geográfica de las regiones de los centros de datos y las regiones de los usuarios.

Para obtener las mediciones de RTT entre cada par de regiones $\langle RCD, RUSR \rangle$ se seleccionaron dos anchors por cada región *RCD* y dos probes por cada región *RUSR*. Dada una región *RCD*, a cada uno de sus dos anchors se le asignó un probe de cada región *RUSR*. En China solo se usó un probe y un anchor por región, debido a la escasa cantidad de anchors y probes presentes en esa zona.

Para seleccionar los anchors y los probes se utilizó la información histórica proporcionada por la plataforma RIPE Atlas respecto a la disponibilidad. Se asumió que la disponibilidad de cada nodo durante los tres meses previos al inicio de las mediciones es un indicador de la disponibilidad de cada nodo durante el período de medición.

En función de la cantidad de créditos disponibles en la plataforma RIPE Atlas y el costo de cada medición, se diseñó un plan para realizar las mediciones de manera que se ejecutaran cada dos horas entre el período comprendido entre las fechas 05/11/2016 y 27/12/2016. Estas mediciones se complementaron con mediciones hechas desde la fecha 03/02/2017 hasta la fecha 10/02/2017.

4.2. Metodología utilizada en el análisis de los datos de las mediciones de RTT

Los datos de las mediciones se descargaron en formato JSON desde la plataforma RIPE Atlas. El procesamiento de los datos se hizo utilizando el lenguaje de programación R y la librería Pandas (McKinney et al., 2010) del lenguaje de programación Python. Luego de obtener todas las mediciones, se hicieron estudios estadísticos de los datos del RTT entre cada par de regiones $\langle RCD, RUSR \rangle$. También se eliminaron los outliers del conjunto de datos. En promedio se obtuvieron aproximadamente 3000 valores de RTT por cada par de $\langle RCD, RUSR \rangle$.

Usando el lenguaje de programación R, se ejecutó el test de normalidad de Kolmogorov-Smirnov (KS) siguiendo la metodología propuesta por Ekstrom (2017). El test de KS para cada conjunto de datos retornó p-valores menores a 2.2×10^{-16}, con lo que se puede inferir que los conjuntos de datos recolectados para las mediciones de RTT no siguen una distribución normal. Por este motivo, se decidió usar la mediana de cada conjunto de datos como estimador de su valor, y como métrica de dispersión el Rango Intercuartílico (Gravetter, 2009).

Para cada región *RUSR* se calculó la mediana del RTT hacia todas las regiones *RCD*, obteniéndose un único valor que representa una estimación de la QoS global disponible para esa región. De forma análoga se calculó la mediana del RTT desde cada región *RCD* hacia todas las regiones *RUSR*. Para el primer caso se usaron en promedio 35000 mediciones por cada par de regiones, mientras que en el segundo caso un promedio de 100000.

Utilizando la librería de Python llamada GeoPandas (Jordahl et al., 2020) se calculó la distancia geográfica en línea recta desde cada región *RCD* hacia cada una de las regiones *RUSR*. Se comparó el valor de la distancia geográfica y el valor de la mediana de RTT entre las regiones *RUSR* y *RCD*.

4.3. Descripción de resultados

En esta sección se reportan los valor de la medianas del RTT para las distintas regiones y los agrupamiento de regiones estudiados. Se compara el valor de la mediana del RTT con el de la distancia geográfica para algunos casos particulares.

Para comparar los valores de RTT entre distintas regiones, se utilizaron gráficos de caja y bigotes (Maindonald, 2010), que son una forma de representar la distribución de los datos. Cada gráfico consiste en una caja con dos bigotes que se extienden desde sus extremos. La caja contiene el 50 % central de los datos, es decir, el rango intercuartílico (IQR), que se calcula como $Q3 - Q1$, donde $Q3$ y $Q1$ son el tercer y primer cuartil, respectivamente. Dentro de la caja, una línea horizontal indica la mediana o segundo cuartil, que divide los datos en dos mitades iguales. Los bigotes marcan los límites del rango de los datos que no se consideran atípicos, según el criterio propuesto por Spatz

(2010): un valor es atípico si es mayor que $Q3 + 1.5 \times IQR$ o menor que $Q1 - 1.5 \times IQR$. Los valores atípicos se muestran como puntos separados del gráfico y pueden deberse a causas extrañas, casos extremos o errores de medición o registro.

4.3.1. QoS global por tipo de región geográfica

Se definió una QoS global desde el punto de vista del usuario y otra QoS desde el punto de vista del proveedor del servicio alojado en cada centro de datos. Se agruparon las mediciones por tipos de regiones y se calculó la mediana y su rango intercuartílico.

QoS global por región del tipo RCD

La QoS global por región RCD que se define como la mediana del RTT entre cada región RCD a todas las regiones $RUSR$. La Figura 4.2 reporta el valor de la mediana del RTT junto al rango intercurtílico para cada centro de datos. Se observa que US East tiene el menor valor de mediana de RTT (117,53 ms), mientras que el mayor valor de la mediana lo tiene Beijing (310,56 ms). En la figura también se puede apreciar la magnitud del rango intercuartílico del RTT para cada centro de datos. El menor valor del rango intercuartílico lo presenta US West (103,14 ms) y el mayor valor lo presenta EU Frankfurt (182,32 ms).

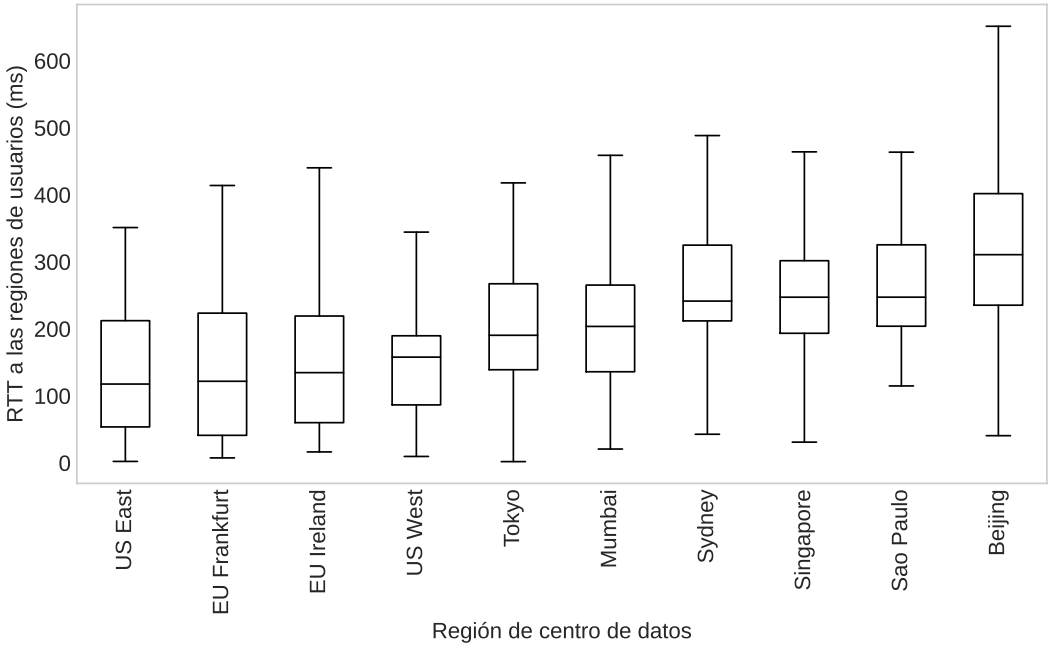

Figura 4.2: Diagrama de caja con bigotes del RTT medido entre cada región de los centros de datos y todas las regiones de usuarios

QoS global por región del tipo RUSR

la QoS global por región ***RUSR*** que se define como: la mediana del RTT desde cada región ***RUSR*** a todas las regiones ***RCD***. La Figura 4.3 reporta el valor de la mediana del RTT junto al rango intercurtílico para cada región de usuarios. Se observa que Georgia tiene el menor valor de mediana de RTT (127,83 ms), mientras que el mayor valor de la mediana lo tiene South Africa (329,53 ms). En la figura también se puede apreciar la magnitud del rango intercuartílico del RTT para cada región de usuario. El menor valor del rango intercuartílico lo presenta Texas (82,41 ms) y el mayor valor lo presenta China (203,81 ms).

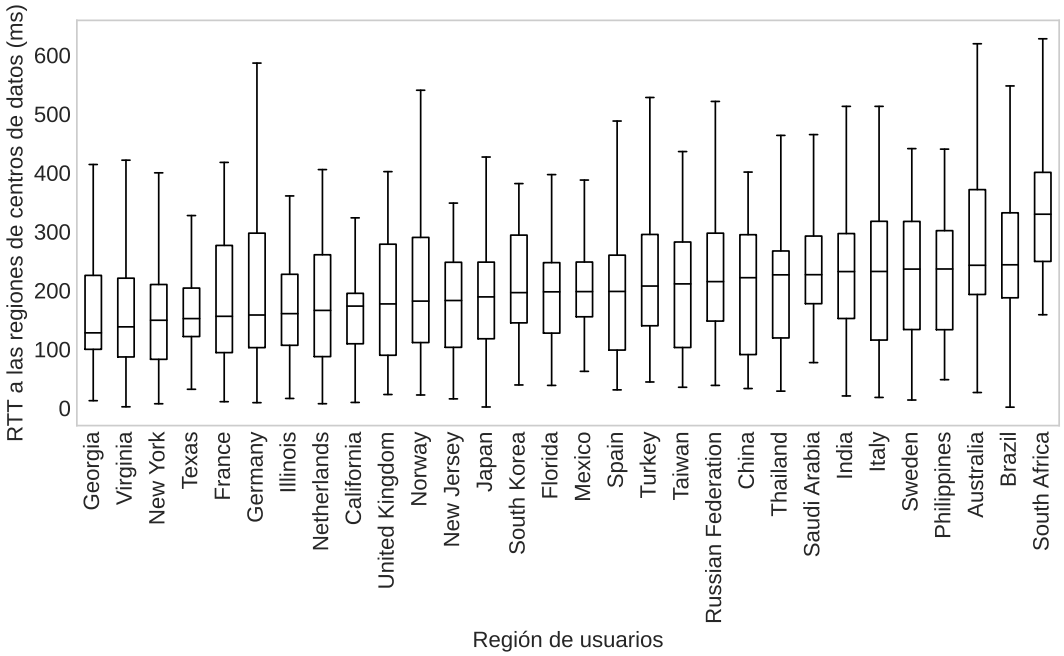

Figura 4.3: Diagrama de caja con bigotes del RTT medido entre cada región de usuarios y todas las regiones de los centros de datos

4.3.2. QoS entre par de regiones RCD y RUSR

La QoS entre un par de regiones ***RCD*** y ***RUSR*** se define como la mediana del RTT medido entre la región ***RCD*** y la región ***RUSR***. Este agrupamiento de mediciones es más detallado que los agrupamientos según región ***RCD*** o ***RUSR***. Por lo tanto, resulta más útil utilizar este agrupamiento de medidas en el modelo del problema planteado en este proyecto de grado.

El menor valor de mediana se da entre las regiones Sao Pablo y Brazil con un valor de 2.00 ms. Por el contrario, el mayor valor de mediana se da entre las regiones de Beijing y South Africa con un valor de 569,98 ms. En cuanto al rango intercuartílico se tiene que el menor rango intercuartílico calculado es de 36,52 ms y el mayor rango intercuartílico calculado es de 452,75 ms.

4.3.3. Comparación entre distancia geográfica y RTT

En esta sección se compara la distancia geográfica y el valor del RTT entre regiones *RCD* y *RUSR*. En la Figura 4.4 se presentan 8 casos particulares de la relación entre el RTT y la distancia geográfica para las regiones.

En la Figura 4.4 se observa que para el caso de US East los resultados se asemejan a una relación lineal positiva entre el RTT y la distancia geográfica. Al aumentar la distancia entre las regiones *RUSR* y el centro de datos en US East el RTT aumenta de manera proporcional. En el caso de EU Frankfurt se puede apreciar que la relación entre el RTT y la distancia es menos dispersa que en US East, pero también se observa una tendencia lineal. Por lo que se puede concluir un comportamiento similar al inferido para US East, pero con menos variabilidad de los datos. Por último, para los casos de Singapore, Sydney y Mumbai los datos muestran una gran dispersión sin una tendencia clara. Mientras que para Tokyo, Beijing y Sao Paulo se observa una dispersión considerable, con algunos puntos indicando una posible relación lineal.

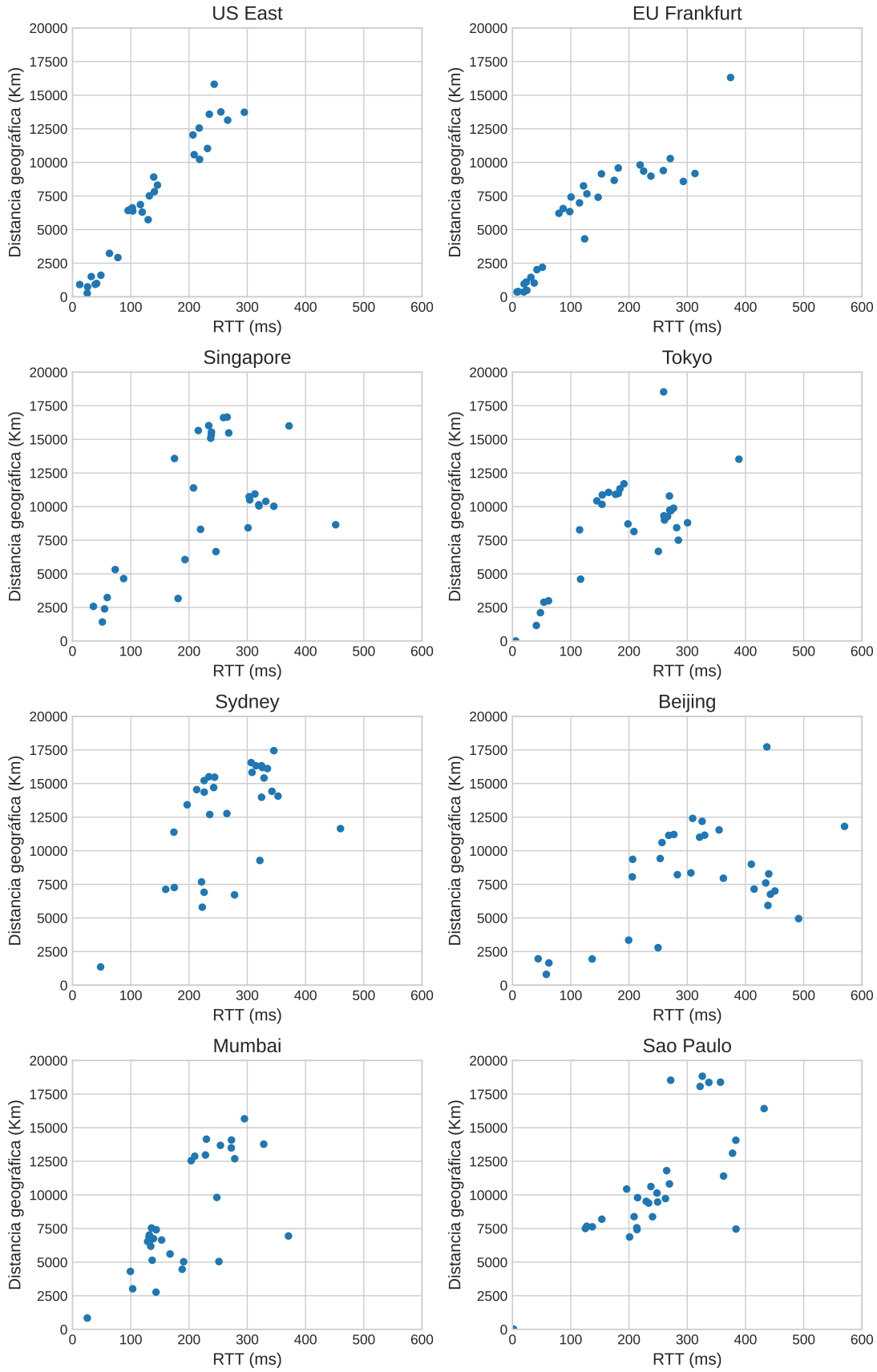

Figura 4.4: Relación entre distancia geográfica y RTT entre las regiones de los centros de datos y las regiones de los usuarios.

Cálculo de una regresión lineal para los valores de distancia y RTT

Utilizando la librería statsmodels (Seabold and Perktold, 2010) de Python se construye un modelo para la aproximación que utiliza la técnica de mínimos cuadrados.Del modelo construido se obtienen los parámetros Intercept y RTT. El parámetro Intercept indica donde la recta de mínimos cuadrados intercepta al eje de la Y de la gráfica correspondiente a la distancia geográfica. El valor del parámetro $RTT = 29.897648$ indica que cuando el RTT varía en 1 milisegundo la distancia varía 29.897648 kilómetros.

Los coeficientes Intercept y RTT se utilizan para construir la fórmula de mínimos cuadrados para predicciones usando el modelo representado en la Ecuación 4.1

$$Y = 2615.819957 + 29.897648 \times X \tag{4.1}$$

En la Figura 4.5 se muestran los valores de la mediana del RTT según la distancia geográfica junto a la recta de mínimos cuadrados determinada por el modelo calculado. Para graficar la línea de mínimos cuadrados se utiliza en la variable X los valores de mínimo y máximo para la mediana del RTT y con esos valores se calcula en la Ecuación 4.1 los valores de la distancia geográfica predicha (valor de Y).

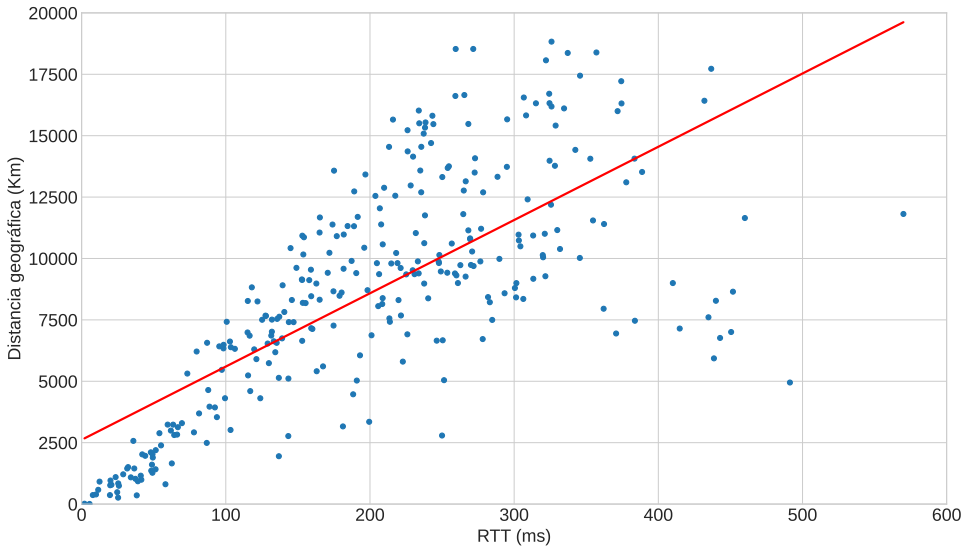

Figura 4.5: Distancia geográfica según valores de la mediana del RTT junto con la recta de mínimos cuadrados calculada.

Resultados de la comparación entre RTT y distancia geográfica

En algunas regiones como US East, EU Frankfurt y Sao Paulo, se observa una relación lineal entre RTT y distancia geográfica, lo que indica para estos casos que el RTT está influenciado principalmente por la distancia geográfica. Sin embargo, en las regiones Singapur, Tokyo, Sydney, Beijing y Mumbai, la dispersión de los datos sugiere que existen otros factores significativos que afectan el valor del RTT. Por lo que se puede concluir que el valor del RTT representa una medida de la QoS más ajustada a la realidad que la que representa la distancia geográfica entre las regiones.

Capítulo 5

Descripción de las técnicas de resolución del problema

Este capítulo presenta las técnicas utilizadas para la resolución del problema de diseño de redes de contenido en plataformas cloud: el AE, las técnicas greedys, la técnica Round Robin y la técnica exacta implementada con *AMPL*.

Para el AE propuesto, en la Sección 5.3 se describen sus operadores evolutivos, la manera en la que se construye su población inicial, la forma de corregir las soluciones y se describe la implementación del AE. Se muestra y describe el algoritmo de enrutamiento de pedidos de recursos de contenido encargado de asignar cada pedido de contenido a una máquina virtual en un centro de datos partiendo de la asignación de recursos determinada por el AE.

En la Sección 5.4 se describen las técnicas Greedy en costo, Greedy en QoS y la técnica Round Robin, que se utilizan para comparar las soluciones calculadas por el AE en la etapa de evaluación experimental. Finalmente, la Sección 5.5 presenta y explica el modelo del problema en *AMPL*.

5.1. Problemas de optimización multiobjetivo

Los problemas de optimización multiobjetivo (Multiobjective Optimization Problems, MOP por su sigla en inglés) son los que proponen la optimización de más de una función objetivo que usualmente están en conflicto entre sí. El hecho de que existan múltiples funciones objetivos implica que en los MOP no existe una solución global única para el problema de optimización, como en el caso de los problemas monoobjetivo, sino un conjunto de soluciones que representan diferentes compromisos entre los valores de las funciones para optimizar (Deb, 2001). La Ecuación 5.1 muestra el modelo matemático general de un MOP.

$$
\begin{aligned}
& min/max\ f_m(x), m = 1, 2..., M \\
& \quad s.a. \\
& \qquad g_s(x) \geqslant 0,\ s = 1, 2, ..., S \\
& \qquad h_r(x) = 0,\ r = 1, 2, ..., R \\
& \qquad x_i^L \leq x_i \leq x_i^U\ 1 \leq i \leq N
\end{aligned}
\tag{5.1}
$$

La solución del MOP es un vector de N variables de decisión $X = (x_1, x_2, ..., x_N)^T$ que debe satisfacer las restricciones impuestas por las funciones $g_s(x), s = 1, 2, ..., S$ y $h_r(x), r = 1, 2, ..., R$ y además debe ofrecer valores de compromiso adecuados para las funciones $f_m(x), m = 1, 2..., M$.

Es necesario definir el concepto de *dominancia*, dado que se utiliza en la mayoría de los algoritmos para optimización multiobjetivo. Una solución X^1 domina a otra solución X^2 si: a) para todos los objetivos la solución X^1 no es peor que la solución X^2 y b) al menos para un objetivo la solución X^1 es mejor que la solución X^2. Cuando se tiene un conjunto finito de soluciones P se puede determinar un subconjunto P' de soluciones no dominadas entre sí al comparar todos los pares posibles de soluciones. Ninguna de las soluciones de P domina a alguna solución de P'. En el caso en que el conjunto P es el conjunto de soluciones factibles del problema, el conjunto P' es el llamado *conjunto óptimo de Pareto*.

El conjunto de los valores obtenidos al aplicar las funciones $f_m(x), m = 1, 2..., M$ al *conjunto óptimo de Pareto* es el *frente de Pareto* del problema de optimización. En la Figura 5.1 se ejemplifica, para un problema de optimización en el que se busca minimizar dos funciones objetivo, las ideas de frente de pareto, soluciones no dominadas y soluciones dominadas.

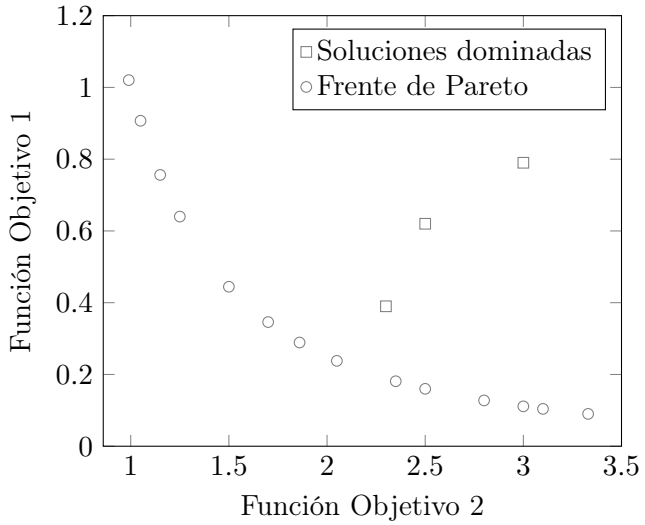

Figura 5.1: Frente de Pareto, soluciones dominadas y soluciones no dominadas.

5.2. Algoritmos evolutivos para optimización multiobjetivo

Las metaheurísticas permiten diseñar técnicas eficientes para encontrar soluciones aproximadas para problemas de optimización, búsqueda y aprendizaje; son estrategias de resolución de problemas de alto nivel que permiten diseñar métodos computacionales para resolver problemas complejos (Glover, 1986). Las metaheurísticas suelen emular fenómenos físicos o naturales, como en el caso de los AE emulan la evolución de las especies en la naturaleza.

Los AE son métodos metaheurísticos no deterministas que resultan útiles para resolver problemas multiobjetivo ya que hallan en cada ejecución un conjunto de soluciones que aproximan el frente de pareto del problema. Los AE generalmente son menos sensibles que otros métodos de búsqueda a la forma del frente de pareto del problema debido a la utilización de operadores evolutivos estocásticos (Deb, 2001).

La Figura 5.2 muestra de forma gráfica dos finalidades que hay que tener en cuenta al diseñar un AE para optimización multiobjetivo (Multiobjective Evolutionary Algorithm, MOEA por su sigla en inglés). El MOEA al mismo tiempo deberá proporcionar un conjunto de soluciones diferentes sin converger a una solución única o a una sección específica del frente de Pareto para mantener la diversidad y también deberá aproximar al frente de Pareto del problema para lograr la convergencia. La convergencia se logra gracias a la utilización de la búsqueda evolutiva basada en Pareto y la diversidad se logra por el uso de técnicas que también se utilizan en la optimización de funciones multimodales (crowding, niches, sharing, etc.).

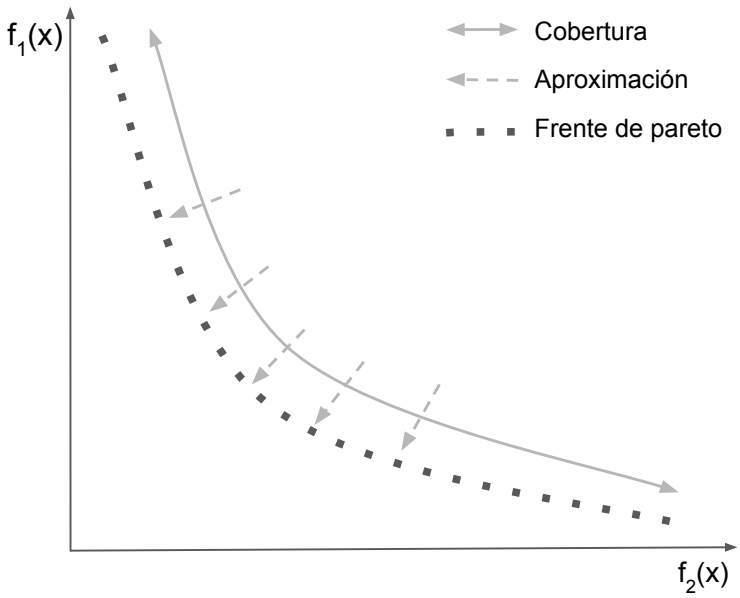

Figura 5.2: Representación gráfica de la diversidad de soluciones del MOEA mediante la cobertura y de la convergencia de las soluciones del MOEA como la aproximación.

5.3. AE para el diseño de redes de contenido en plataformas cloud

En esta sección se describen aspectos generales del algoritmo NSGA-II utilizado para resolver el problema. Se describe el procedimiento de construcción de la población inicial del AE y se detallan los operadores evolutivos construidos de forma específica para resolver el problema planteado.

5.3.1. Modelo de resolución del problema en tres fases mediante el uso del algoritmo evolutivo NSGA-II

Para resolver el problema de diseño de redes de distribución de contenidos en plataformas cloud se implementó un AE del tipo NSGA-II. El algoritmo NSGA-II (Non-dominated Sorting Genetic Algorithm II) es un MOEA que fue propuesto por Deb et al. (2000) como una versión mejorada de su antecesor NSGA (Srinivas and Deb, 1994). El algoritmo NSGA-II incorporó las siguientes características que le permiten tener una búsqueda evolutiva mejorada respecto al algoritmo NSGA: i) un ordenamiento elitista no–dominado, el cual utiliza una subpoblación auxiliar, que disminuye la complejidad asociada a la comprobación de dominancia; ii) un procedimiento de asignación de fitness, basado en rangos de no dominación, que incorpora los valores de distancia de crowding para evaluar la diversidad de las soluciones; iii) el mecanismo para la preservación de la diversidad de la población utiliza una técnica de crowding que no requiere parámetros adicionales como sí requiere la técnica de sharing usada en su predecesor NSGA.

El AE del tipo NSGA-II implementado tiene tres fases: inicialización de la población, planificación y enrutamiento. La Figura 5.3 muestra las actividades principales que componen a cada fase del AE. Las fases de inicialización de la población y la de planificación determinan la asignación de máquinas virtuales a los centros de datos para poder cumplir con los pedidos de contenidos. La fase de enrutamiento determina cuál es la máquina virtual que atiende cada pedido de contenido. Las soluciones del AE son de la forma <S-Planificación,S-Enrutamiento> donde *S-Planificación* se construye en las primeras dos fases y la componente *S-Enrutamiento* se construye en la fase de enrutamiento.

Primera fase: Inicialización de la población inicial del AE

La fase de inicialización de la población del AE se ejecuta una vez al comienzo del algoritmo. La población inicial del AE consiste en soluciones que tienen la forma <S-Planificación , $\emptyset$ >. Las componentes *S-Planificación* de las soluciones iniciales del AE se eligen de manera aleatoria a partir de un conjunto de soluciones G, utilizando una distribución uniforme. El conjunto G se construye con soluciones obtenidas a través de las heurísticas Greedy en costo y Greedy en QoS, que se detallan en la Sección 5.4. A cada solución seleccionada se le aplica el operador de mutación con probabilidad *pMutarSolucionInicial* con la finalidad de generar diversidad genética en la población inicial.

Segunda fase: Planificación

En esta fase el AE determina para cada centro de datos la cantidad de cada tipo de máquinas virtuales asignadas en cada hora de planificación. También el AE determina cuáles recursos de contenido se almacenan en cada centro de datos. En esta fase el AE construye soluciones de la forma <S-Planificación , $\emptyset$ >. A cada solución se le aplican los operadores de cruzamiento y mutación con sus respectivas probabilidades.

Tercera fase: Enrutamiento

En esta fase el AE determina cuál es el centro de datos que atiende cada pedido de recurso de contenido. En la fase de enrutamiento, para cada solución, se construye la componente S-Enrutamiento utilizando el algoritmo de enrutamiento de pedidos de recursos de contenido que se describe en la Sección 5.3.6.

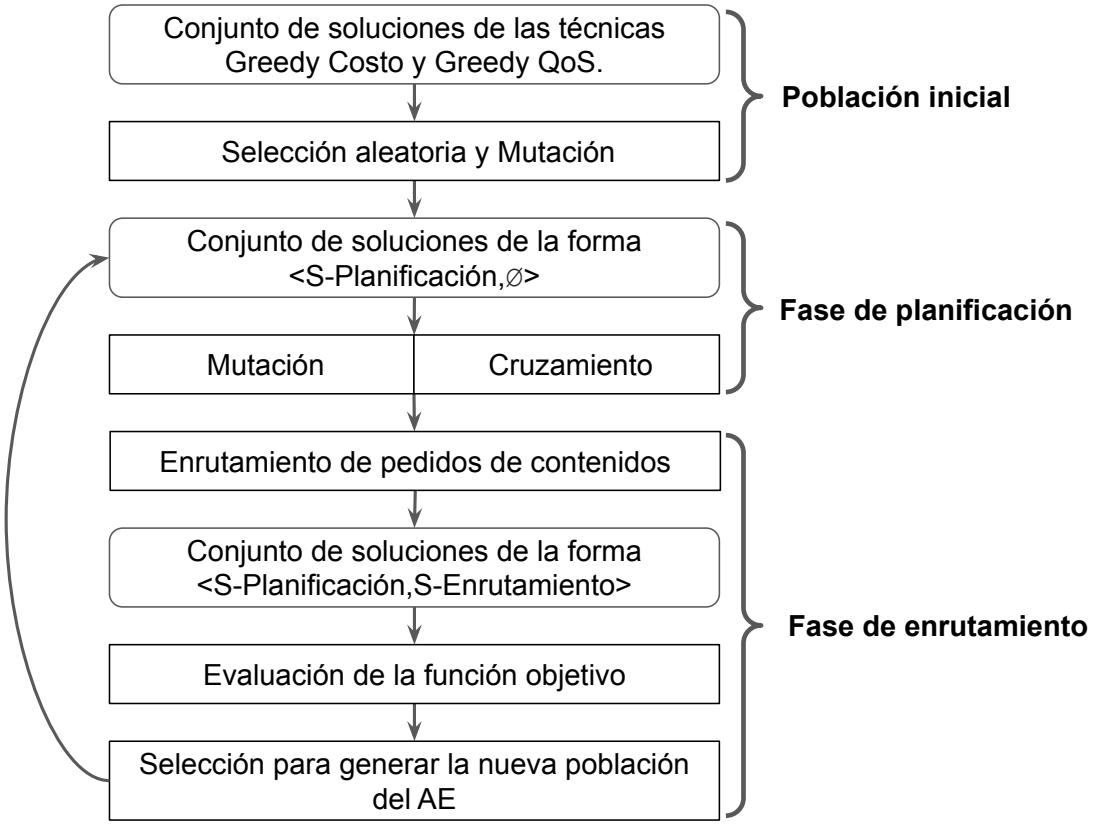

Figura 5.3: Fases del algoritmo evolutivo implementado para resolver el problema de diseño de redes de distribución de contenidos en plataformas cloud.

Finalmente, cuando cada solución de la población del AE tiene sus dos componentes, <S-Planificación,S-Enrutamiento>, se evalúan las funciones objetivo sobre la población. Mientras no se cumpla el criterio de parada del AE, luego de evaluar las funciones objetivo el AE selecciona las soluciones que forman parte de la nueva población y vuelve al comienzo de la fase de planificación.

5.3.2. Representación de las soluciones

En el algoritmo NSGA-II propuesto las soluciones son representadas por matrices:

- Y, matriz de números enteros de dimensión $H \times m$.
- X, matriz binaria de dimensión $n \times m$.
- $\tilde{Y}$, matriz de números enteros de dimensión $m \times 1$.
- Z, matriz de números enteros de dimensión $s \times n \times m \times v \times (H \times 60)$.

H es la cantidad de horas a planificar en el período $[0, T]$, m es la cantidad de centros de datos, n es la cantidad de recursos de contenidos, s es cantidad de regiones geográficas desde donde los usuarios hacen los pedidos y v es la cantidad de máquinas virtuales asignadas en todo el periodo de planificación.

La componente *S-Planificación* se representa por las matrices Y, $\tilde{Y}$ y X. La matriz $\tilde{Y}$ indica cuantas instancias de máquinas virtuales reservadas tiene asignadas cada centro de datos durante todo el período de planificación. En la matriz X se representa la asignación de los recursos de proveedores de contenidos que tiene cada centro de datos. La matriz

Y indica para cada centro de datos la cantidad de máquinas virtuales que son necesarias para cubrir la demanda de pedidos de recursos de contenidos en cada hora de planificación $h \in [0, H-1]$. La cantidad de instancias de máquinas virtuales que se solicitan a demanda se puede expresar para cada hora $h \in [0, H-1]$ y para cada centro de datos c_e de la siguiente manera: $\max\{0, y_e^h - \tilde{y}_e\}$.

La componente *S-Enrutamiento* se representa por la matriz Z. El valor $Z[l, i, e, v, t] = b$ indica que los usuarios de la región rl descargan un total b de veces el contenido k_i desde la máquina virtual v del centro de datos c_e en el instante de tiempo t (en esta matriz el tiempo se expresa en minutos).

La Ecuación 5.2 muestra para una hora h del período de planificación de $[0, H-1]$, la codificación de la componente *S-Planificación* de una posible solución para una instancia con tres centros de datos (C_0, C_1, C_2) y seis recursos (K_0, ..., K_5). La Tabla 5.1 muestra la asignación de recursos de contenido a los centros de datos y las cantidades de máquinas virtuales utilizadas en cada centro de datos en el ejemplo planteado.

Tabla 5.1: Máquinas virtuales y recursos asignados a cada dentro de datos en el ejemplo de representación de la componente *S-Planificación* de la solución.

Centros de datos	$\tilde{Y}$	Y^h	$\max\{0, Y^h - \tilde{Y}\}$	Recursos asignados
C_0	3	5	2	K_0, K_3
C_1	3	3	0	K_1, K_2, K_4
C_2	5	4	0	K_0, K_5, K_2

$$\tilde{Y} = \begin{pmatrix} 3 \\ 3 \\ 5 \end{pmatrix} \begin{matrix} C_0 \\ C_1 \\ C_2 \end{matrix}$$

$$X = \begin{matrix} \\ K_0 \\ K_1 \\ K_2 \\ K_3 \\ K_4 \\ K_5 \end{matrix} \begin{matrix} \begin{matrix} C_0 & C_1 & C_2 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{matrix} \tag{5.2}$$

$$Y = \begin{matrix} \\ C_0 \\ C_1 \\ C_2 \end{matrix} \begin{matrix} \begin{matrix} 0 & \cdots & h & \cdots & H-1 \end{matrix} \\ \begin{pmatrix} \cdots & \cdots & 5 & \cdots & \cdots \\ \cdots & \cdots & 3 & \cdots & \cdots \\ \cdots & \cdots & 4 & \cdots & \cdots \end{pmatrix} \end{matrix}$$

5.3.3. Operador de cruzamiento

El operador de cruzamiento está basado en el cruzamiento uniforme con adaptaciones para contemplar las restricciones del problema. Este operador recibe dos soluciones factibles, realiza el cruzamiento sobre la componente *S-Planificación* de cada solución y retorna dos soluciones factibles que son el resultados de cruzar las dos soluciones recibidas.

El operador de cruzamiento intercambia entre las soluciones, para cada centro de datos, la asignación de recursos de contenidos, la asignación total de máquinas virtuales y la proporción de máquinas virtuales que son asignadas a demanda. Cada una de estas categorías es seleccionada de manera aleatoria con distribución uniforme (probabilidad 0,5) para ser intercambiada entre los centros de datos de las soluciones.

El Algoritmo 1 muestra el pseudocódigo del operador de cruzamiento propuesto. Para las dos soluciones que participan del cruzamiento, el primer ciclo del algoritmo (líneas 1–4) intercambia la asignación de recursos de contenidos a los centros de datos. Para cada recurso $k_i \in K$ seleccionado con distribución uniforme (probabilidad 0,5) y para cada centro de datos $c_e \in C$, el ciclo intercambia la asignación del recurso k_i al centro de datos c_e entre las dos soluciones que participan del cruzamiento (línea 4). Como el intercambio descrito se realiza para todos los centros de datos $c_e \in C$ entonces no se generarán soluciones no factibles por falta de asignación de algún recurso.

El segundo ciclo del algoritmo (líneas 5–8) selecciona cada hora de planificación $h \in [0, H]$ con distribución uniforme (probabilidad 0,5) y para cada hora h seleccionada, el ciclo intercambia entre las dos soluciones la asignación de las máquinas virtuales a los centros de datos para todos los centros de datos en C (línea 8). Como el intercambio descrito se realiza para todos los centros de datos $c_e \in C$ entonces no se generarán soluciones no factibles por no poder cubrir la demanda de máquinas virtuales en una hora de la planificación $h \in [0, H]$.

El último ciclo del algoritmo (líneas 9–11) selecciona cada centro de datos $c_e \in C$ con distribución uniforme (probabilidad 0,5) y para cada centro de datos c_e seleccionado, el ciclo intercambia entre las soluciones implicadas en el cruzamiento las asignaciones de máquinas virtuales reservadas en c_e para todo el periodo de planificación (línea 11).

Para todos los centros de datos $c_e \in C$, se debe cumplir que: la cantidad total de máquinas virtuales asignadas siempre sea mayor o igual a la cantidad de máquinas virtuales reservadas. Por lo tanto con los intercambios entre las soluciones que hacen los dos últimos ciclos del algoritmo se pueden generar soluciones que no sean factibles. Para corregir este último caso la línea 12 ejecuta el Algoritmo 3 de corrección de soluciones que se describe en la Subsección 5.3.5.

Algoritmo 1: Operador de cruzamiento

```
1  para cada k_i ∈ K
2    Con probabilidad = 0,5 :
3      para cada c_e ∈ C
4        intercambia x_ie entre las soluciones
5  para cada h ∈ [0, H]
6    Con probabilidad = 0,5 :
7      para cada c_e ∈ C
8        intercambia y_e^h entre las soluciones
9  para cada c_e ∈ C
10   Con probabilidad = 0,5 :
11     intercambia ỹ_e entre las soluciones
12 ejecutar el Algoritmo 3 de corrección de soluciones
```

5.3.4. Operador de mutación

El operador de mutación recibe una solución factible, realiza la mutación sobre la componente *S-Planificación* de la solución y retorna una solución factible. Este operador cambia la asignación de recursos de contenidos a los centros de datos teniendo en cuenta que no queden recursos de contenidos sin asignar. En lo referente a la asignación de máquinas virtuales a los centros de datos, el operador de mutación produce cambios en la solución que hacen variar la proporción de máquinas virtuales reservadas y a demanda que usa el centro de datos para cubrir la demanda de recursos de contenido.

El Algoritmo 2 muestra el pseudocódigo del operador de mutación. El primer ciclo del algoritmo (líneas 1–7) cambia en la solución la asignación de recursos a los centros de datos. Para cada recurso $k_i \in K$, el ciclo selecciona centros de datos $c_e \in C$ con distribución uniforme (probabilidad pM). Si el recurso ya está asignado al centro de datos seleccionado, el ciclo quita la asignación del recurso al centro de datos luego de comprobar que el recurso esté asignado al menos a algún otro centro de datos de C (línea 7). La comprobación de que el recurso esté asignado a otro centro de datos es para evitar generar una solución no factible por falta de asignación de recursos. Si el recurso no está asignado al centro de datos seleccionado, el ciclo asigna el recurso al centro de datos (línea 5).

El segundo ciclo del algoritmo (líneas 8–17) modifica en la solución la asignación de máquinas virtuales a los centros de datos. Para cada hora de planificación $h \in [0, H]$, el ciclo selecciona centros de datos $c_e \in C$ con distribución uniforme (probabilidad pM). Si el centro de datos c_e tiene y_e^h máquinas virtuales asignadas, el ciclo selecciona un valor $cantMV$ con distribución uniforme en el conjunto $[1, y_e^h]$ (línea 12), luego quita la cantidad de $cantMV$ de máquinas virtuales asignadas a ese centro de datos en la hora h (línea 14) y las asigna en la hora h a otro centro de datos C_f seleccionado con probabilidad uniforme (línea 15). Si el centro de datos c_e no tiene máquinas virtuales asignadas, el ciclo le asigna una máquina virtual (línea 17). Como la mutación descrita no modifica la cantidad total de máquinas virtuales asignadas a los centros de datos en cada hora de planificación h, entonces no se generarán soluciones no factibles por no poder cubrir la demanda de máquinas virtuales en la hora h.

El tercer ciclo (líneas 18–23) modifica en la solución la cantidad de máquinas virtuales reservadas en cada centro de datos $c_e \in C$ seleccionado con distribución uniforme (probabilidad pM). Si el centro de datos seleccionado tiene $\tilde{y}_e$ máquinas virtuales reservadas, el ciclo selecciona un valor con distribución uniforme $cantMV \in [0, \tilde{y}_e - 1]$ y asigna ese valor como nuevo valor de cantidad de máquinas virtuales reservadas del centro de datos c_e (línea 21). Si el centro de datos seleccionado no tiene una máquina virtual reservada, el ciclo le asigna una (línea 23).

Modificar la cantidad de máquinas virtuales asignadas y reservadas a un centro de datos $c_e \in C$ puede producir soluciones que no sean factibles, por lo tanto luego de modificar la cantidad de máquinas virtuales asignadas en los centros de datos, en la línea 24, se ejecuta el Algoritmo 3 de corrección de soluciones que se describe en la Subsección 5.3.5.

Algoritmo 2: Operador de mutación

para cada $k_i \in K$
  para cada $c_e \in C$
    Con $probabilidad = pM$:
      si $x_{ie} = 0$ **entonces**
        $x_{ie} = 1$
      si $x_{ie} = 1$ y $\exists$ $c_f \in C / x_{if} = 1$ **entonces**
        $x_{ie} = 0$
para cada $h \in [0, H]$
  para cada $c_e \in C$
    Con $probabilidad = pM$:
      si $y_e^h > 0$ **entonces**
        Seleccionar un valor $cantMV \in [1, y_e^h]$
        Seleccionar un centro de datos $c_f \neq c_e \in C$
        $y_e^h = y_e^h - cantMV$
        $y_f^h = y_f^h + cantMV$
      en otro caso
        $y_e^h = 1$
para cada $c_e \in C$
  Con $probabilidad = pM$:
    si $\tilde{y}_e > 0$ **entonces**
      $\tilde{y}_e = cantMV$ con $cantMV \in [0, \tilde{y}_e - 1]$
    en otro caso
      $\tilde{y}_e = 1$
ejecutar el Algoritmo 3 de corrección de soluciones

5.3.5. Algoritmo de corrección de soluciones

Para todos los centros de datos $c_e \in C$, se debe cumplir que: la cantidad total de máquinas virtuales asignadas siempre debe ser mayor o igual a la cantidad de máquinas virtuales reservadas. Por lo tanto, aplicar el operador de cruzamiento o el operador de mutación puede generar soluciones no factibles. El algoritmo de corrección de soluciones modifica la cantidad de máquinas virtuales reservadas en los centros de datos para que las soluciones cumplan la condición de factibilidad descrita.

El Algoritmo 3 muestra el pseudocódigo del método de corrección de soluciones. El único ciclo del algoritmo (líneas 1–4) modifica la cantidad de máquinas virtuales reservadas en cada centro de datos para que la solución sea factible. Para cada centro de datos $c_e \in C$, el ciclo calcula el valor de mc como la mínima cantidad de máquinas virtuales usadas en el centro de datos en todo el período de planificación (línea 2). Si la cantidad de máquinas reservadas en el centro de datos ($\tilde{Y}_e$) es mayor a mc, el ciclo le asigna al centro de datos mc máquinas virtuales reservadas (línea 4).

Algoritmo 3: Método de corrección de soluciones

```
1 para cada c_e ∈ C
2     mc = mínima cantidad de máquinas virtuales usadas en c_e
3     si ỹ_e > mc entonces
4         ỹ_e = mc
```

5.3.6. Algoritmo de enrutamiento de pedidos de recursos de contenido

El algoritmo de enrutamiento de pedidos de recursos de contenido (AEP) inicia la fase de enrutamiento del AE. Se basa en la asignación de máquinas virtuales y recursos de contenidos especificado en la componente *S-Planificación* de la solución y construye la componente *S-Enrutamiento*. En la componente *S-Enrutamiento* de la solución el AEP asigna cada pedido de recurso de contenido a un centro de datos respetando las restricciones del problema especificadas en la Sección 2.3.2.

Para asignar los pedidos de recursos a los centros de datos el AEP selecciona con probabilidad 0.5 y distribución uniforme entre dos estrategias de asignación de pedidos de recursos: la estrategia que prioriza el costo (*ruteoMejorCosto*) o la estrategia que prioriza QoS (*ruteoMejorQoS*). Una vez seleccionada una estrategia de asignación el AEP la aplica para asignar todos los pedidos de recursos de contenidos.

Las estrategias *ruteoMejorCosto* y *ruteoMejorQoS* tienen más de un método greedy de asignación de recursos de contenidos. Si un método no logra asignar todos los recursos de contenidos, se descartan las asignaciones hechas hasta el momento y se aplica el siguiente método que tiene restricciones más relajadas. En cada estrategia los métodos se ordenan desde el más restrictivo hasta el menos restrictivo, de esta manera los métodos quedan ordenados en forma decreciente en función de la calidad (medida según costo o QoS) de la solución que construyen.

Estrategia de enrutamiento que prioriza costo

La estrategia *ruteoMejorCosto* tiene dos métodos de asignación de pedidos de recursos de contenido: *m1Costo* y *m2Costo*. Ambos métodos seleccionan la máquina virtual más barata que cumple con las condiciones impuesta por el método.

El método *m1Costo* es el más restrictivo de la estrategia, tiene como condición necesaria para asignar un pedido de contenido que el centro de datos tenga alojado el recurso de contenido correspondiente al pedido. A los efectos de optimizar el uso de máquinas virtuales el método *m1Costo*, para cada instante de tiempo, prioriza a las máquinas virtuales que ya están siendo usadas por el mismo proveedor de contenido que el que tiene el recurso asociado al pedido. En caso de no encontrar una máquina virtual usada por el proveedor mencionado, se selecciona una máquina virtual libre.

El método *m2Costo* consiste en aplicar el método *m1Costo* pero sin la restricción que exige que la máquina virtual a la que se le asigna el pedido esté en un centro de datos que tenga asignado el recurso de contenido asociado al pedido. Este último método no respeta la asignación de recursos a centros de datos representada en la componente *S-Planificación* de la solución, por lo tanto la estrategia *ruteoMejorCosto* debe corregir la componente *S-Planificación* al aplicar el método *m2Costo*.

El Algoritmo 4 muestra el pseudocódigo de la estrategia *ruteoMejorCosto* del algoritmo AEP. La estrategia *ruteoMejorCosto* utiliza la función ***obtenerMvMejorCosto***$(t, p, restriccion\_actual)$ para buscar en cada instante de tiempo t una máquina virtual con el mejor costo que cumpla con las restricciones a), b) y c) del problema (Sección 2.3.2) y además con la restricción adicional *restriccion_ actual*.

El primer ciclo del algoritmo (líneas 1–12) implementa el método de asignación *m1Costo*. El método *m1Costo* de manera iterativa para cada pedido de contenido en cada instante de tiempo t, busca una máquina virtual de mejor costo aplicando la función ***obtenerMvMejorCosto***$(t, p, restriccion\_actual)$ (línea 5). La restricción *restriccion_ actual* impone que la máquina virtual retornada por la función esté siendo usada por el proveedor de contenidos asociado al pedido que atenderá dicha máquina virtual, de manera adicional la restricción impone que el centro de datos tenga asignado el recurso asociado al pedido. Si la función no encontró una máquina virtual que cumpla la condición, entonces se repite la búsqueda (línea 8) quitando de la condición *restriccion_ actual* la exigencia de que la máquina virtual retornada por la función esté siendo usada por el proveedor de contenidos asociado al pedido. Si se encontró una máquina virtual en una de las dos búsquedas hechas, entonces se asocia el pedido a la máquina virtual encontrada. Si no se encontró una máquina virtual en las búsquedas hechas, entonces termina el método *m1Costo* y a continuación se ejecuta el método *m2Costo*.

El segundo ciclo del algoritmo (líneas 13–23) implementa el *m2Costo*. Al aplicar el método *m2Costo* no se tiene en cuenta ninguna de las asignaciones de pedidos de contenidos hechas en el *m1Costo*, el proceso de asignación comienza nuevamente desde el primer pedido. Al igual que el método *m1Costo* el método *m2Costo* itera sobre los pedidos de recursos de contenidos para cada instante de tiempo t. La primera búsqueda de una máquina virtual para cada pedido hecha por el método *m2Costo* invoca a la función ***obtenerMvMejorCosto***$(t, p, restriccion\_actual)$ (línea 19) con *restriccion_ actual* exigiendo que la máquina virtual retornada por la función esté siendo usada por el proveedor de contenidos asociado al pedido, en este caso la función podrá retornar una máquina virtual que esté en un centro de datos que no tenga asignado el recurso de contenido. Si la función no encontró una máquina virtual tal que cumpla la condición impuesta, entonces el método *m2Costo* busca una máquina virtual sin una *restriccion_ actual* asociada, la máquina virtual solo cumplirá las restricción del problemas presentes en la función ***obtenerMvMejorCosto***.

El método *m2Costo* siempre encuentra una máquina virtual para cada pedido de

contendido (lo garantiza la factibilidad de la componente de la solución *S-Planificación*), por lo tanto en la última línea del algoritmo se asigna el pedido a la máquina virtual encontrada y se asigna el recurso de contenido asociado al pedido al centro de datos de la máquina virtual si ya no estaba asignado.

Algoritmo 4: Algoritmo de enrutamiento de pedidos de recursos de contenido priorizando costo

```
   // Método m1Costo
 1 para cada t ∈ [0,T]
 2   para cada p ∈ Pedidos(t)
 3     k_p = recurso de p, pro_k = proveedor de k_p
 4     restriccion_actual = el centro de datos tiene el recurso k_p y pro_k usando mv
 5     mv =obtenerMvMejorCosto(t, p, restriccion_actual)
 6     si no encontró mv entonces
 7       restriccion_actual = el centro de datos tiene el recurso k_p
 8       mv =obtenerMvMejorCosto(t, p, restriccion_actual)
 9     si encontró mv entonces
10       asigna pedido de p a mv
11     si no encontró mv entonces
12       fin Método m1Costo
13 si hay algún p ∈ Pedidos(t) sin asignar entonces
14   // Método m2Costo
15   para cada t ∈ [0,T]
16     para cada p ∈ Pedidos(t)
17       k_p = recurso de p, pro_k = proveedor de k_p
18       restriccion_actual = pro_k usando mv
19       mv =obtenerMvMejorCosto(t, p, restriccion_actual)
20       si no encontró mv entonces
21         restriccion_actual = ∅
22         mv =obtenerMvMejorCosto(t, p, restriccion_actual)
23       asigna pedido de p a mv y asigna k_p al centro de datos de mv si no estaba
     asignado
```

Estrategia de enrutamiento que prioriza QoS

La estrategia *ruteoMejorQoS* tiene cuatro métodos de asignación de pedidos de recursos de contenido: *m1QoS*, *m2QoS*, *m3QoS* y *m4QoS*. El método *m1QoS* cuando asigna un pedido de contenido a una máquina virtual aplica como restricción que la máquina virtual esté en un centro de datos que tenga asignado el recurso de contenido asociado al pedido. El método *m2QoS* integra al *m1QoS* la restricción de darle prioridad a las máquinas virtuales que ya estén siendo usadas por el proveedor de contenidos asociado al pedido. Los métodos *m3QoS* y *m4QoS* son el resultado de quitarle a los métodos *m1QoS* y *m2QoS* la restricción que exige que la máquina virtual a la que se le asigna el pedido esté en un centro de datos que tenga asignado el recurso de contenido asociado al pedido.

Los métodos *m3QoS* y *m4QoS* no respetan la asignación de recursos a los centros de datos representada en la componente *S-Planificación* de la solución, por lo tanto la estrategia *ruteoMejorQoS* debe corregir dicha componente en el caso de asignar el pedido

de un recurso de contenido a un centro de datos que no tenga el recurso asociado.

El Algoritmo 5 muestra el pseudocódigo de la estrategia *ruteoMejorQoS* del algoritmo AEP. La estrategia *ruteoMejorQoS* utiliza en cada uno de sus métodos la función ***obtenerMvMejorQoS***(*t, p, restriccion_ actual*) para buscar en cada instante de tiempo t una máquina virtual con el mejor QoS que cumpla con las restricciones a), b) y c) del problema (Sección 2.3.2) y además con la restricción adicional *restriccion_ actual* la cual varía según el método empleado.

El primer ciclo del algoritmo (líneas 1–9) implementa el método de asignación *m1QoS*. El método *m1QoS* de manera iterativa para cada pedido de contenido en cada instante de tiempo t, busca una máquina virtual de mejor QoS aplicando la función ***obtenerMvMejorQoS***($t, p, restriccion\_actual$) (línea 5). La restricción *restriccion_ actual* impone que la máquina virtual retornada por la función pertenezca a un centro de datos que tenga asignado el recurso asociado al pedido. Si la función encontró una máquina virtual, se asocia el pedido a la máquina virtual encontrada. Si la función no encontró una máquina virtual, termina el método *m1QoS* y a continuación se ejecuta el método *m2QoS*.

El segundo ciclo del algoritmo (líneas 12–23) implementa el método *m2QoS*. El método *m2QoS* no tiene en cuenta ninguna de las asignaciones de pedidos de contenidos hechas en el *m1QoS*, el proceso de asignación comienza nuevamente desde el primer pedido. Al igual que el método *m1QoS* el método *m2QoS* itera sobre los pedidos de recursos de contenidos para cada instante de tiempo t. El método *m2QoS* realiza la primera búsqueda de una máquina virtual de mejor QoS aplicando la función ***obtenerMvMejorQoS***($t, p, restriccion\_actual$) (línea 16). La restricción *restriccion_ actual* impone que la máquina virtual retornada por la función esté siendo usada por el proveedor de contenidos asociado al pedido que atenderá dicha máquina virtual, de manera adicional la restricción impone que el centro de datos tenga asignado el recurso asociado al pedido. Si la función no encontró una máquina virtual que cumpla la condición, entonces se repite la búsqueda (línea 19) quitando de la condición *restriccion_ actual* la exigencia de que la máquina virtual retornada por la función esté siendo usada por el proveedor de contenidos asociado al pedido. Si se encontró una máquina virtual en una de las dos búsquedas hechas, entonces se asocia el pedido a la máquina virtual encontrada. Si no se encontró una máquina virtual en las búsquedas hechas, entonces termina el método *m2QoS* y a continuación se ejecuta el método *m3QoS*.

El método *m3QoS* tiene la misma estructura que el método *m1QoS* con dos diferencias. La primera diferencia es que la invocación a la función ***obtenerMvMejorQoS***($t, p, restriccion\_actual$) (línea 30) la realiza quitando en la *restriccion_ actual* la exigencia de que el centro de datos tenga asignado el recurso asociado al pedido. La segunda diferencia es que el método *m3QoS* modifica la componente *S-Planificación* de la solución (asigna el recurso al centro de datos) en el caso de asignar el pedido de un recurso de contenido a un centro de datos que no tenga el recurso asociado (línea 32). De forma análoga el método *m4QoS* tiene la misma estructura que el método *m2QoS* con dos diferencias. La primera diferencia es que las invocaciones a la función ***obtenerMvMejorQoS***($t, p, restriccion\_actual$) en las líneas 41 y 44 la realiza quitando en la *restriccion_ actual* la exigencia de que el centro de datos tenga asignado el recurso asociado al pedido. La segunda diferencia es que el método *m4QoS* modifica la componente *S-Planificación* de la solución (asigna el recurso al centro de datos) en el caso de asignar el pedido de un recurso de contenido a un centro de datos que no tenga el recurso asociado. (línea 45).

Algoritmo 5: Algoritmo de enrutamiento de pedidos de recursos de contenido priorizando QoS

```
   // Método m1QoS
 1 para cada t ∈ [0,T]
 2   para cada p ∈ Pedidos(t)
 3     k_p = recurso de p
 4     restriccion_actual = el centro de datos tiene el recurso k_p
 5     mv =obtenerMvMejorQoS(t, p, restriccion_actual)
 6     si encontró mv entonces
 7       asigna pedido p a mv
 8     si no encontró mv entonces
 9       fin Método m1QoS
10 si hay algún p ∈ Pedidos(t) sin asignar entonces
11   // Método m2QoS
12   para cada t ∈ [0,T]
13     para cada p ∈ Pedidos(t)
14       k_p = recurso de p, pro_k = proveedor de k_p
15       restriccion_actual = pro_k usando mv y el centro de datos tiene el recurso k_p
16       mv =obtenerMvMejorQoS(t, p, restriccion_actual)
17       si no encontró mv entonces
18         restriccion_actual = el centro de datos tiene el recurso k_p
19         mv =obtenerMvMejorQoS(t, p, restriccion_actual)
20       si encontró mv entonces
21         asigna pedido p a mv
22       si no encontró mv entonces
23         fin Método m2QoS
24   si hay algún p ∈ Pedidos(t) sin asignar entonces
25     // Método m3QoS
26     para cada t ∈ [0,T]
27       para cada p ∈ Pedidos(t)
28         k_p = recurso de p
29         restriccion_actual = ∅
30         mv =obtenerMvMejorQoS(t, p, restriccion_actual)
31         si encontró mv entonces
32           asignar pedido de p a mv y asignar k_p al centro de datos de mv
33         si no encontró mv entonces
34           fin Método m3QoS
35     si hay algún p ∈ Pedidos(t) sin asignar entonces
36       // Método m4QoS
37       para cada t ∈ [0,T]
38         para cada p ∈ Pedidos(t)
39           k_p = recurso de p, pro_k = proveedor de k_p
40           restriccion_actual = pro_k usando mv
41           mv =obtenerMvMejorQoS(t, p, restriccion_actual)
42           si no encontró mv entonces
43             restriccion_actual = ∅
44             mv =obtenerMvMejorQoS(t, p, restriccion_actual)
45           asignar pedido de p a mv y asignar k_p al centro de datos de mv
```

5.3.7. Implementación del AE

El AE fue implementado utilizando el framework JMetal (Durillo and Nebro, 2011). JMetal tiene como objetivo el desarrollo, la experimentación y el estudio de metaheurísticas para resolver problemas de optimización multiobjetivo. Jmetal está desarrollado en Java usando el paradigma de orientación a objetos. En JMetal se tiene a disposición un conjunto de optimizadores clásicos y modernos, un conjunto de problemas de referencia y un conjunto de indicadores de calidad para evaluar el rendimiento de los algoritmos. Con JMetal se pueden configurar y ejecutar estudios experimentales completos que a su vez generan información estadística de los resultados obtenidos.

5.4. Heurísticas de base para la comparación

En esta sección se presentan las heurísticas Greedy en costo, Greedy en QoS y Round Robin. Los resultados de estas heurísticas son utilizados para la comparación con los resultados del AE propuesto.

Para poder hacer la comparación es necesario generar un frente de pareto con las soluciones de cada heurística. Por lo tanto, para cada heurística, se genera un conjunto de soluciones: *conjuntoSolucionesGreedyCosto*, *conjuntoSolucionesGreedyQoS* y *conjuntoSolucionesRoundRobin*. A cada uno de estos conjuntos se le eliminan las soluciones dominadas para generar el frente de pareto de cada heurística. Los conjuntos *conjuntoSolucionesGreedyCosto*, *conjuntoSolucionesGreedyQoS* también son utilizados para generar la población inicial del AE.

Cuando en un instante de tiempo t es necesario asignar una nueva máquina virtual a un centro de datos, las tres heurísticas utilizan el concepto de *pico de demanda* de máquinas virtuales (*cantMVpico*) para determinar el tipo de máquina virtual asignada. Si en el instante de tiempo t se da un *cantMVpico*, las heurísticas determinan que el tipo de máquina virtual asignada es contratada a demanda para la hora a la que pertenece t. Si, por el contrario, en el instante de tiempo t no se da un *cantMVpico*, las heurísticas determinan que se renta una nueva máquina virtual por todo el período de planificación.

Para generar diversidad de soluciones en cada heurística se varía el valor del *cantMVpico* y de esta manera utilizando la función *calcularInstantesDeTiempoPico*(*cantMVpico*) se determinan los instantes de tiempo donde la cantidad de máquinas virtuales necesarias para cubrir la demanda es mayor o igual a *cantMVpico*. Cada heurística genera un conjunto de soluciones con distinta cantidad y proporción de tipos de máquina virtuales asignadas a los centros de datos en cada instante de tiempo.

El valor de *cantMVpico* varía en el intervalo comprendido entre cero y el valor máximo de las cantidades mínimas de máquinas virtuales que son necesarias para cubrir la demanda en cada instante de tiempo. Para calcular la demanda mínima de máquinas virtuales para cada hora de planificación, en función de los pedidos de recursos, se tiene en cuenta la restricción de que en un mismo instante de tiempo no pueden haber dos proveedores usando una misma máquina virtual y además que una máquina virtual no puede atender más de CR pedidos en simultáneo.

5.4.1. Greedy en costo

La heurística Greedy en costo asigna de manera iterativa a las máquinas virtuales los pedidos de recursos de contenido para cada instante de tiempo de planificación. Para cada pedido de contenido, la heurística selecciona una máquina virtual en un centro de datos de mejor costo que a su vez cumple un umbral mínimo de QoS. Si para un pedido de recurso de contenido la heurística no encuentra una máquina virtual que pueda atender al pedido, suma una nueva máquina virtual usada en el centro de datos seleccionado. Como el tipo de máquina virtual agregado depende de si en el instante de tiempo t hay un *pico de demanda* los instantes de tiempo marcados con picos de demanda y el umbral mínimo de QoS son parámetros del algoritmo.

El Algoritmo 6 muestra el pseudocódigo de la heurística Greddy en costo. La heurística consiste en un ciclo que recorre los pedidos de recursos de contenido para cada instante de tiempo de planificación (líneas 3–21). En el ciclo, para cada pedido de recurso, la heurística determina el centro de datos de mejor costo que a su vez cumple con un mínimo de QoS determinado por la variable *umbralQoS*; luego realiza el incremento de los costos asociados al almacenamiento y a la transferencia de datos en el centro de datos seleccionado (líneas 6–10).

En la línea 11 la función *obtenerMVconRecursosDisponibles* determina, considerando las restricciones del problema, si el centro de datos seleccionado tienen una máquina virtual disponible para atender al pedido de recursos. En caso de encontrar una máquina virtual disponible, asigna el pedido de recurso a la máquina virtual encontrada. Si no encuentra una máquina virtual disponible entonces, determina en función de si instante de tiempo pertenece a una hora con demanda pico, el tipo de máquina virtual que agregará en el centro de datos (líneas 15–19). La heurística asigna en la nueva máquina virtual el pedido de recurso y luego asigna la máquina virtual al centro de datos seleccionado (líneas 20–21).

El algoritmo de construcción del conjunto *conjuntoSolucionesGreedyCosto* varía los parámetros *instantesDeTiempoPico* y *umbralQoS* de la heurística Greedy en costo para generar soluciones distintas. Los parámetros son calculados teniendo en cuenta los pedidos de contenidos en el tiempo, las restricciones del problema y las mediciones de QoS entre regiones de los usuarios y las regiones de los centros de datos.

Algoritmo 6: Heurística Greedy en costo para la asignación de pedidos de recursos y máquinas virtuales a los centros de datos.

```
1  funcion greedyCosto(instantesDeTiempoPico, umbralQoS)
2  inicio
3    para cada t ∈ [0,T]
4      pedidos = obtener pedidos de recursos en t
5      para cada p ∈ pedidos
6        centroDeDatos = más barato que cumple umbralQoS según la región de p
7        asignar recurso de contenido de p a centroDeDatos
8        incrementar métrica de QoS según centroDeDatos y la región de p
9        incrementar costo de almacenamiento según centroDeDatos
10       incrementar costo de transferencia según centroDeDatos
11       mvDisponible = obtenerMVconRecursosDisponibles(p, centroDeDatos)
12       si mvDisponible no es null entonces
13         asignar p a mvDisponible
14       en otro caso
15         si t ∈ instantesDeTiempoPico entonces
16           h = hora de planificación a la que pertenece t
17           mvDisponible = rentar máquina a demanda en la hora h
18         en otro caso
19           mvDisponible = rentar máquina reservada durante el período [0,T]
20         asignar p a mvDisponible
21         asignar mvDisponible a centroDeDatos
22 Retornar solución s
23 fin
```

El Algoritmo 7 comienza con la inicialización de variables, en esta sección calcula el máximo requerimiento de máquinas virtuales que se da en algún instante de tiempo del período de planificación (línea 2). También se ordenan los valores de QoS de manera decreciente (línea 3). El ciclo comprendido entre las líneas 5 y 11 itera variando el valor de la variable *cantMVpico* desde 0 hasta la cantidad máxima de máquinas calculadas en la línea 2.

La variable *cantMVpico* se usa como parámetro en la función *calcularInstantesDeTiempoPico* para determinar los instantes de tiempo en donde hay un pico de demanda (línea 6). El ciclo comprendido entre las líneas 7 y 10 recorre los valores de QoS ordenados e invoca la función *greedyCosto* (que implementa al Algoritmo 6) usando como parámetros los instantes de tiempo con demanda pico y el umbral de QoS actual del ciclo. Si la función *greedyCosto* encontró una solución para esos parámetros, agrega esa solución al conjunto *conjuntoSolucionesGreedyCosto*. Finalmente, en la línea 11 se incrementa la variable que indica cuantas máquinas virtuales representan un pico de demanda.

Algoritmo 7: Construcción del conjunto de soluciones halladas con la heurística Greedy en costo.

```
1  cantMVpico =0
2  maximoPicoMVrequeridas = máximo requerimiento de máquinas virtuales en [0,T]
3  conjuntoUmbralQoS = ordenar decreciente valores de QoS
4  conjuntoSolucionesGreedyCosto = ∅
5  mientras cantMVpico <= maximoPicoMVrequeridas
6     instantesDeTiempoPico = calcularInstantesDeTiempoPico(cantMVpico)
7     para cada umbralQoS ∈ conjuntoUmbralQoS
8        s = greedyCosto(instantesDeTiempoPico, umbralQoS)
9        si s no es null entonces
10          agregar s a conjuntoSolucionesGreedyCosto
11    cantMVpico = cantMVpico + 1
```

5.4.2. Greddy en QoS y Round Robin

Las heurísticas Greddy en QoS y Round Robin solo se diferencian entre sí por la manera en la que seleccionan los centros de datos, por lo tanto se puede mostrar su estructura con los mismos algoritmos. Estas dos heurísticas se diferencian de la heurística Greedy en costo, a parte de la manera de seleccionar los centros de datos, en que no utilizan umbrales de QoS al momento de seleccionar centros de datos.

El Algoritmo 8 muestra el pseudocódigo de las heurísticas Greedy en Qos y Round Robin, el algoritmo tiene como parámetro la identificación de la heurística utilizada para poder determinar la manera en la que se seleccionan los centros de datos según el tipo de heurística (líneas 6 a 10). Para la heurística Greddy en QoS se selecciona el centro de datos de mejor QoS (línea 7). Para la heurística Round Robin se numeran los centros de datos y en cada paso se selecciona el centro de datos cuyo identificador es una unidad superior al identificador del centro de datos seleccionado en el la selección anterior. Si el centro de datos de la selección anterior tiene el último identificador disponible, se vuelve al identificador del primer centro de datos. Por esta razón se utilizar la función Modulo matemático con la cantidad de centros de datos (línea 10). El resto de la estructura del algoritmo (línea 11 hasta el final) es igual al Algoritmo 6 para la heurística de Greedy en

costo.

El algoritmo de construcción de los conjuntos *conjuntoSolucionesGreedyQoS* y *conjuntoSolucionesRoundRobin* varía el parámetro *instantesDeTiempoPico* de la heurística Greedy en costo y Round Robin para generar soluciones distintas. El parámetro es calculado teniendo en cuenta los pedidos de contenidos en el tiempo, las restricciones del problema y las mediciones de QoS entre regiones de los usuarios y las regiones de los centros de datos.

Algoritmo 8: Estructura de las heurística Greedy en QoS y Round Robin para la asignación de pedidos de recursos y máquinas virtuales a los centros de datos.

```
1  funcion greedyQoS_RoundRobin(instantesDeTiempoPico, tipoHeuristica)
2  inicio
3    para cada t ∈ [0,T]
4      pedidos =obtener pedidos de recursos en t
5      para cada p ∈ pedidos
6        si tipoHeuristica es GreedyQoS entonces
7          centroDeDatos = centro de datos con mejor QoS según la región de p
8        si tipoHeuristica es RoundRobin entonces
9          centroDeDatos =último centro de datos seleccionado +1
10         centroDeDatos = Modulo(cantCentrosDatos, centroDeDatos)
11       asignar recurso de contenido de p a centroDeDatos
12       incrementar métrica de QoS según centroDeDatos y la región de p
13       incrementar costo de almacenamiento según centroDeDatos
14       incrementar costo de transferencia según centroDeDatos
15       mvDisponible = obtenerMVconRecursosDisponibles(p, centroDeDatos)
16       si mvDisponible no es null entonces
17         asignar p a mvDisponible
18       en otro caso
19         si t ∈ instantesDeTiempoPico entonces
20           h = hora de planificación a la que pertenece t
21           mvDisponible = rentar máquina a demanda en la hora h
22         en otro caso
23           mvDisponible = rentar máquina reservada durante el período [0,T]
24         asignar p a mvDisponible
25         asignar mvDisponible a centroDeDatos
26 Retornar solución s
27 fin
```

El Algoritmo 9 muestra el pseudocódigo de la construcción del conjunto de soluciones obtenidas con las heurísticas Greedy en costo y Round Robin. En la primera línea del algoritmo se determina la heurística empleada. Este algoritmo se diferencia con el Algoritmo 7 de la heurística Greddy en costo en que para generar diversidad de soluciones no utiliza el conjunto de medidas de QoS, solo utiliza la variación de cantidad de máquinas virtuales consideradas pico de demanda.

Algoritmo 9: Construcción del conjunto de soluciones, halladas con la heurística Greedy en QoS o Round Robin, que se utiliza para comparar con las soluciones encontradas por el algoritmo evolutivo.

```
1  tipoHeuristica = GreedyQoS o RoundRobin
2  cantMVpico =0
3  maximoPicoMVrequeridas = máximo requerimiento de máquinas virtuales en [0,T]
4  conjuntoSoluciones = ∅
5  mientras cantMVpico <= maximoPicoMVrequeridas
6     instantesDeTiempoPico = calcularInstantesDeTiempoPico(cantMVpico)
7     s = greedyQoS_RoundRobin(instantesDeTiempoPico, tipoHeuristica)
8     agregar s a conjuntoSoluciones
9     cantMVpico = cantMVpico + 1
10 si tipoHeuristica es GreedyQoS entonces
11    conjuntoSolucionesGreedyQoS = conjuntoSoluciones
12 si tipoHeuristica es RoundRobin entonces
13    conjuntoSolucionesRoundRobin = conjuntoSoluciones
```

5.5. Resolución del problema de forma exacta con AMPL

En esta sección se presenta la resolución del problema utilizando la herramienta AMPL (Fourer et al., 2003) para resolver el modelo de forma exacta. La sección se organiza describiendo AMPL y CPLEX (CPLEX Optimizer) y luego presentando el modelo implementado.

5.5.1. Herramientas usadas para resolver el problema de forma exacta

Para modelar y resolver el problema de diseño de redes de contenido en plataformas cloud de forma exacta se utilizó AMPL, un lenguaje de modelado algebraico para programación matemática que permite expresar problemas de optimización tales como los problemas de programación lineal.

AMPL se caracteriza por tener expresiones algebraicas y representación de conjuntos semejantes a las usadas en la notación matemática habitual, por lo que la representación del modelo del problema a resolver en este lenguaje es similar al modelo expresado en notación matemática. Además, AMPL permite separar el modelo del problema de los datos específicos para cada instancia del problema. AMPL ofrece un entorno de comando interactivo que permite configurar y resolver los problemas de programación matemática expresados con el lenguaje.

La Figura 5.4 muestra un esquema de la manera en la que interaccionan con el entorno de comando los principales componentes utilizados por AMPL para resolver el problema. Las indicaciones al intérprete del lenguaje AMPL se almacenen en un archivo con extensión .run, allí se indica el archivo donde se encuentra la expresión del modelo (archivo .mod), el archivo donde están los datos para la instancia del problema a resolver (archivo .dat) y se indica el optimizador con el cual se resolverá el problema. Para resolver el problema de diseño de CDN en plataformas cloud el optimizador utilizado fue CPLEX (CPLEX Optimizer).

CPLEX puede resolver problemas de programación lineal, programación entera mixta

y programación cuadrática. El el problema de diseño de CDN en plataformas cloud se clasifica como un problema de programación entra mixta ya que solo tiene variables enteras y variables binarias. CPLEX para resolver este tipo de problemas utiliza branch and cut (Branch and Cut in CPLEX; Wolsey, 2020).

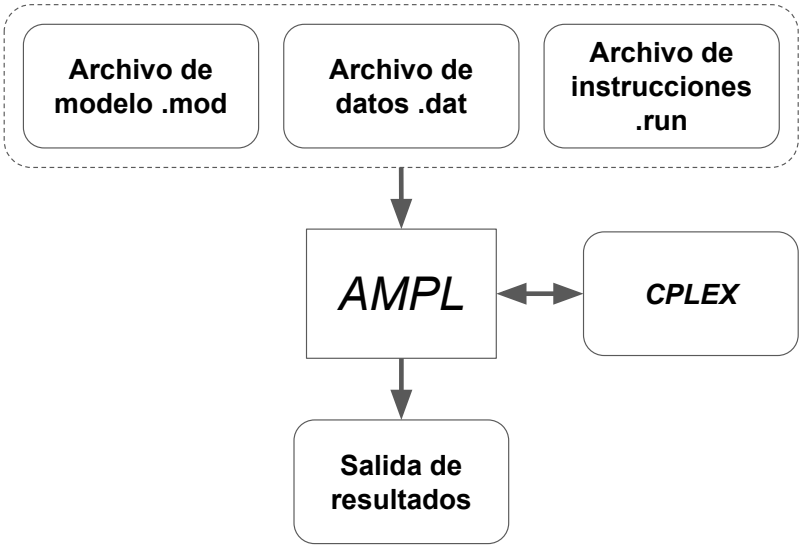

Figura 5.4: Diagrama del funcionamiento de AMPL-CPLEX

5.5.2. Presentación del modelo implementado en AMPL

El problema de diseño de CDN en plataformas cloud se modeló como un problema de optimización mono-objetivo, en el cual se minimiza el costo de construir una CDN sujeto a una restricción que indica que la métrica de QoS no supere un umbral dado por el valor de *umbralQoS*. Al variar el parámetro *umbralQoS* de manera decreciente se incrementa la dificultad para cumplir la restricción de QoS lo cual genera soluciones cuyo costo aumenta al variar el parámetro. Con este conjunto de soluciones generado se construye un frente de Pareto.

El parámetro *umbralQoS* se inicializa con un valor igual a la cantidad de pedidos de recursos multiplicado por el peor valor (el mayor) de QoS entre una región de usuario y un centro de datos. Luego el parámetro varía en el conjunto de enteros que tiene como máximo el valor inicial calculado para *umbralQoS* y como valor inferior 1.

El Algoritmo 10 presenta el código *AMPL* que expresa los parámetros, conjuntos y variables del modelo del problema. La línea 9 declara el parámetro $picoMaximoRequestInteger$ que se calcula como la mayor cantidad de pedidos de recursos en un minuto de la planificación. El conjunto *demandaset* declarado en la línea 16 se utiliza para modelar de forma eficiente la demanda de recursos k durante el tiempo de planificación desde distintas regiones de usuarios. El parámetro $precursosProveedor$ indica a que proveedor pertenece cada recurso. El parámetro $cCostoReservadas$ indica el precio por hora de una máquina virtual reservada en cada centro de datos mientras que el parámetro $cCostoAdemanda$ indica el precio por hora de una máquina virtual a demanda en cada centro de datos.

Finalmente, el parámetro *qQoS* almacena la métrica de calidad de servicio entre cada región de usuarios y cada centro de datos.

Las variables del problema se declaran a partir de la línea 27. La primera variable declarada es *yReservadas* que representa, como vector de enteros, cuantas máquinas virtuales reservadas se usan en cada centro de datos. La variable *yUsadas* es una matriz de enteros que representa cuantas máquinas virtuales totales se usan en cada centro de datos en cada hora de la planificación. La variable *xMapeorecursos* es una matriz de binarios que representa en que centros de datos está guardado cada recurso. La variable *zRuteo* es una matriz de enteros que almacena la información sobre qué centro de datos atiende cada pedido de recurso y por último, la variable *vmCantVms* es una matriz de enteros que almacena la cantidad de máquinas virtuales que usa cada proveedor en cada centro de datos para cada minuto de la planificación.

El Algoritmo 11 presenta la función objetivo y las restricciones del problema modelados en *AMPL*. En la línea 1 se define la función objetivo *Fcost* que se calcula como la suma del costo de transferencia total de todos los recursos (líneas 2 y 3) más el costo del almacenamiento total de los recursos en los centros de datos (líneas 4 y 5) más el costo total de las máquinas virtuales reservadas (líneas 6 y 7) más la suma del costo total de las máquinas a demanda (líneas 8 y 9). La cantidad de máquinas a demanda usadas se calcula como la diferencia entre la cantidad de máquinas usadas y la cantidad de máquinas reservadas.

En las líneas 10 a 12 se define la restricción *QoS*, que impone que la suma total de las métricas de QoS entre los centros de datos con pedidos asignados y la región de usuario de donde proviene cada pedido sea menor o igual al parámetro *umbralQoS*. La restricción *PisoUsadasReservadas* definida en las líneas 13 a 15 impone, para cada centro de datos y cada hora de planificación, la condición de que no puede haber más máquinas reservadas que las usadas. Las restricciones *UnProvVMyCapacidadDeVM* (líneas 16 a 19) y *UnProvVMyCapacidadDeVMAux* (líneas 20 a 24) se usan de forma conjunta para asegurar que una máquina virtual solo pueda ser usada por un proveedor a la vez en cada instante de tiempo y que cada máquina virtual no exceda su capacidad de atender hasta *CR* pedidos de recursos en cada instante de tiempo, respectivamente. La restricción *EstaVideoEnDC* (líneas 25 a 27) indica que si se asigna un pedido de recurso k a un centro de datos entonces el centro de datos tiene el recurso k asignado. Por último la restricción *CubrirDemandaSet* es usada para asegurar que se cubre la demanda de pedidos de recursos.

Algoritmo 10: Parámetros, conjuntos y variables del problema modelado en AMPL

```
1  param CR                                >= 0;
2  param cantdcs                        integer >= 0;
3  param umbralQoS                         >= 0;
4  param cantrecursos                   integer >= 0;
5  param cantproveedores                integer >= 0;
6  param tiempoTotalHoras               integer >= 0;
7  param picoMaximoRequest                 >= 0;
8  param cantRegionesUsuarios           integer >= 0;
9  param picoMaximoRequestInteger       integer >= 0;
10 set K                              = 1..cantrecursos;
11 set Thoras                         = 1..tiempoTotalHoras;
12 set Tminutos                       = 1..tiempoTotalHoras * 60;
13 set PROVEEDORES                    = 1..cantproveedores;
14 set REGIONESUSR                    = 1..cantRegionesUsuarios;
15 set posibleDemandasPositivas       = 1..picoMaximoRequestInteger;
16 set demandaset within K
17                    cross Tminutos
18                    cross REGIONESUSR
19                    cross posibleDemandasPositivas;
20 set CENTROSDATOS                   = 1..cantdcs;
21 param precursosProveedor {K, PROVEEDORES} binary;
22 param cCostoReservadas {CENTROSDATOS} >= 0;
23 param cCostoAdemanda {CENTROSDATOS} >= 0;
24 param cCostoTransferirCadarecurso {CENTROSDATOS} >= 0;
25 param cCostoAlmacenarCadarecurso {CENTROSDATOS} >= 0;
26 param qQos {CENTROSDATOS, REGIONESUSR} >= 0;
27 var yReservadas {CENTROSDATOS} integer >= 0;
28 var yUsadas {CENTROSDATOS, Thoras} integer >= 0;
29 var xMapeorecursos {CENTROSDATOS, K} binary;
30 var zRuteo {CENTROSDATOS, K, Tminutos, REGIONESUSR} integer >= 0;
31 var vmCantVms {PROVEEDORES, CENTROSDATOS, Tminutos} integer >= 0;
```

Algoritmo 11: Función objetivo y restricciones del problema modelado en AMPL

```
minimize Fcost :
  sum t in Tminutos, c in CENTROSDATOS, v in K, r in REGIONESUSR
   zRuteo[c, v, t, r] * cCostoTransferirCadarecurso[c] +
  sum v in K, c in CENTROSDATOS
   xMapeorecursos[c, v] * cCostoAlmacenarCadarecurso[c] +
  sum c in CENTROSDATOS
   yReservadas[c] * tiempoTotalHoras * cCostoReservadas[c] +
  sum t in Thoras, c in CENTROSDATOS
   (yUsadas[c, t] - yReservadas[c]) * cCostoAdemanda[c];
subject to Qos :
sum t in Tminutos, c in CENTROSDATOS, v in K, r in REGIONESUSR
  zRuteo[c, v, t, r] * qQos[c, r] <= umbralQoS;
subject to PisoUsadasReservadas
  t in Thoras, c in CENTROSDATOS :
   yReservadas[c] <= yUsadas[c, t];
subject to UnProvVMyCapacidadDeVM
  tminutos in Tminutos, c in CENTROSDATOS :
   (sum p in PROVEEDORES vmCantVms [p, c, tminutos])
    <= yUsadas[c, ceil(tminutos/60)];
subject to UnProvVMyCapacidadDeVMAux
  tminutos in Tminutos, c in CENTROSDATOS, p in PROVEEDORES :
   (sum v in K, r in REGIONESUSR
    precursosProveedor[v, p] * zRuteo[c, v, tminutos, r])/CR
     <= vmCantVms [p, c, tminutos];
subject to EstarecursoEnDC
 v in K, r in REGIONESUSR, c in CENTROSDATOS, t in Tminutos :
 (zRuteo[c, v, t, r] / picoMaximoRequest) <= xMapeorecursos[c, v];
subject to CubrirDemandaSet (v, t, r, d) in demandaSet :
 sum c in CENTROSDATOS zRuteo[c, v, t, r] = d;
```

Capítulo 6

Análisis experimental

En este capítulo se describe la metodología utilizada para el análisis experimental de las heurísticas empleadas para resolver el problema de diseño de redes de contenido en plataformas cloud. Se reportan los resultados obtenidos. Se muestran las características de las instancias utilizadas y la manera en la que se generaron y usaron. Se describen las principales características de la plataforma de desarrollo y ejecución. Se presenta la metodología utilizada para obtener la mejor configuración de parámetros del AE utilizado. Se reportan los resultados obtenidos al comparar las soluciones del AE con las soluciones obtenidas con las distintas heurísticas utilizadas para resolver el problema.

6.1. Instancias del problema

Para configurar y evaluar el AE propuesto, se realizaron tres etapas: ajuste de parámetros, evaluación experimental y resolución del problema de forma exacta. Para cada etapa se generó un conjunto de instancias realistas del problema.

Para la generación de las instancias se utilizó la herramienta GlobeTraff desarrollada por Katsaros et al. (2012). GlobeTraff es un generador de tráfico que permite crear mezclas realistas de tráfico de Internet. GlobeTraff puede generar varios tipos de tráfico de aplicaciones basándose en modelos presentes en la literatura, permite la parametrización detallada de los modelos generados y la composición de la mezcla de tráfico resultante.

En cada instancia generada al tráfico generado por GlobeTraff se lo dividió por regiones geográficas. Para hacer una división realista se usó información proporcionada en el reporte realizado por la empresa Cisco (2016), en el cual se plantea una división del tráfico por tipos y zonas geográficas.

Para el ajuste de parámetros del AE, se generaron escenarios variando el tamaño de las simulaciones de tráfico hechas con GlobeTraff en función del tiempo de ejecución y la cantidad de ejecuciones independientes del AE. Para la evaluación experimental del AE se ajustó el tamaño de las instancias en tres grupos: chicas, medianas y grandes. Para ajustar el tamaño de las instancias de la evaluación experimental se usaron como referencia datos de cantidad de pedidos de videos en el tiempo obtenidos en estadísticas del proyecto openfing ((OpenFING), 2017).

Para todas las instancias generadas se mantuvo fija la cantidad de regiones de usuarios en 30 y la cantidad de centros de datos en 10 (como se explica en el Capítulo 4), el resto de las características se muestran en la Tabla 6.1.

Tabla 6.1: Características de las instancias generadas para las etapas: Ajuste de parámetros del AE, Evaluación experimental del AE y Resolución del problema con CPLEX.

Etapa	Instancia	#videos	#minutos	#proveedores	#pedidos de videos
Ajuste de parámetros del AE	A1	500	300	6	134
	C1	1000	240	7	364
	D1	400	300	5	249
	D2	4000	480	7	600
Evaluación experimental del AE	CH1	11500	240	6	8219
	CH2	11500	240	6	6744
	M1	18600	240	6	13000
	M2	18600	240	6	12304
	G1	30000	240	6	27700
	G2	30000	240	6	22827
Resolución del problema con CPLEX	AMPL1	500	240	6	48
	AMPL2	300	300	6	303
	AMPL3	400	300	5	249
	AMPL4	400	420	5	547

6.2. Metodología utilizada en la evaluación experimental

En esta sección se presenta la metodología utilizada en la evaluación experimental del AE. Se describe la metodología utilizada en el ajuste de parámetros del AE y luego se describe la metodología utilizada para comparar el AE con las técnicas de referencia.

En la evaluación experimental se reportarán los valores de: mejor solución respecto a costo, mejor solución respecto a QoS, solución de compromiso, hipervolúmen relativo (HVR) y tiempo de ejecución. Para determinar la solución de compromiso se selecciona la solución que se encuentran más cercana al cero según la norma Euclídea. La métrica HVR asociado a una heurística es el cociente entre el volumen determinado por el frente de Pareto calculado para la heurística y el volumen determinado por el frente de Pareto de referencia. El valor ideal del cociente es 1.

6.2.1. Metodología utilizada en el ajuste de parámetros del AE

Debido a la naturaleza estocástica de los AE, es necesario efectuar un conjunto de de ejecuciones independientes (variando la semilla del generador de números aleatorios utilizado) que permitan determinar la significancia estadística de los resultados obtenidos utilizando diferentes configuraciones de parámetros. Los experimentos se realizaron sobre las instancias $A1$, $C1$, $D1$ y $D2$. Para cada instancia se realizaron 50 ejecuciones independientes con criterio de parada de 1000 generaciones para cada combinación de los parámetros probabilidad de cruzamiento (pC), probabilidad de mutación (pM) y tamaño de población ($\#P$). A los parámetros se les asignaron los siguientes valores candidato: $pC \in \{0,5; 0,7; 0,9\}$; $pM \in \{0,001; 0,01; 0,1\}$ y $\#P \in \{50; 100; 200\}$.

Dada la cantidad de ejecuciones necesarias para determinar la mejor combinación de parámetros, se seleccionó el tamaño de las instancias para que el tiempo de ejecución total del experimento fuese razonable para esa etapa del procedimiento. Los valores máximos

usados fueron: 4000 videos, 480 minutos planificados, 7 proveedores y 600 pedidos de videos.

En función de los *p*-valores obtenidos mediante la prueba de Kolmogorov-Smirnov (K-S) se determinó que no todas las instancias siguieron distribución normal para las métricas del HVR y el tiempo de ejecución del AE. Por lo tanto, se optó por la utilización de la mediana de las métricas como estimador de los resultados.

Para las cuatro instancias usadas (A1, C1, D1 y D2) se hizo una clasificación de cada combinación de parámetros según su valor de mediana de HVR. En los casos donde la prueba no paramétrica de Kruskal-Wallis no permitió determinar significancia estadística con un nivel de confianza del 95 %, que permita asegurar una diferencia significativa entre la distribución de resultados de HVR de dos configuraciones paramétricas, se utilizó el tiempo de ejecución del AE para determinar las posiciones relativas en la clasificación. Se determinó, para cada combinación de parámetros, las veces que ocupó cada posición en las distintas clasificaciones. La finalidad de la clasificación fue determinar los menores valores de cada métrica: HVR y tiempo.

6.2.2. Metodología utilizada en la comparación del AE con las técnicas de referencia

En la evaluación experimental se resolvió el problema para diez instancias. Para cuatro instancias el problema se resolvió de forma exacta utilizando CPLEX y para todas las instancias se resolvió el problema con las técnicas: AE, Greedy costo, Greedy QoS y Round Robin.

Para cada instancia del problema se compararon los resultados obtenidos por el AE con el resto de las técnicas de resolución del problema. Se compararon las mejores soluciones respecto a costo y a QoS, también se compararon las soluciones de compromiso. Para el caso de las técnicas CPLEX, AE y Greedy costo se compararon los valores del HVR a un frente de Pareto de referencia.

Dependiendo de la técnica empleada para resolver el problema se utilizaron distintos métodos para generar conjuntos de soluciones. En el caso del AE, dado que se trata de una técnica estocástica, para cada instancia se realizaron 50 ejecuciones independientes. En cada ejecución la librería utilizada para resolver el AE genera un frente de Pareto. En el caso de las técnicas Greedy costo, Greedy QoS y Round Robin; se variaron parámetros de cada técnica para generar conjuntos de soluciones, para cada técnica se eliminaron las soluciones dominadas para generar un frente de Pareto. Para Greedy QoS y Round Robin al eliminar las soluciones dominadas el conjunto resultante fue de un número tan chico de soluciones que no es posible construir un frente de Pareto. En la resolución del problema de forma exacta con CPLEX se generaron las soluciones resolviendo múltiples problemas monobjetivo. Cada problema monobjetivo optimizó el costo mientras que se usó un valor de QoS como límite de una restricción de QoS. Al usar un conjunto de valores de QoS distintos, se generó un conjunto de soluciones calculadas con CPLEX.

Para cada instancia del problema se generó un frente de Pareto de referencia juntando las soluciones de todas las técnicas y eliminando del conjunto las soluciones dominadas. Utilizando el frente de Pareto de referencia se calculó el HVR de cada técnica para las cuales se pudo construir al menos un frente de Pareto.

En función de los *p*-valores obtenidos mediante la prueba de K-S se determinó que para el caso del AE los valores de HVR no siguieron distribución normal. Por lo tanto, se optó por la utilización de la mediana como estimador del HVR del AE. Para las técnicas

Greedy costo y CPLEX, que tienen solo un frente de parteto, se calculó un único valor de HVR.

Como en el caso del AE se utiliza la mediana del HVR (HVRmed) y en el resto de técnicas determinísticas se utiliza un único valor del HVR es necesario asegurar la significancia estadísticas de las comparaciones. Por esta razón es necesario tener en cuenta el valor del rango intercuartílico (RIQ) al comparar la mediana del HVR del AE (HVRmed(AE)) con el valor del HVR de cada técnica T (HVRmed(T)).

Para asegurar la significacincia estadística de la diferencia entre los valores del HVR de las técnicas a comparar se tiene que cumplir que: el valor absoluto de las diferencias de las HVRmed sea mayor al máximo valor de los respectivos RIQ (Ecuación 6.1).

$$|HVRmed(AE) - HVRmed(T)| > max(RIQ(HVR(AE), RIQ(HVR(T)) \tag{6.1}$$

Como la técnica T es determinística se cumple lo que se plantea en la Ecuación 6.2.

$$\begin{aligned} HVRmed(T) &= HVR(T) \\ RIQ(HVR(T)) &= 0 \end{aligned} \tag{6.2}$$

Por lo tanto si se cumple la desigualdad planteada en la Ecuación 6.3 se puede concluir que la diferencia entre el valor del HVR del AE y el valor del HVR de la técnica T es estadísticamente significativa.

$$|HVRmed(AE) - HVR(T)| > RIQ(HVR(AE)) \tag{6.3}$$

6.3. Plataforma de desarrollo y ejecución

Para el desarrollo de las heurísticas se utilizó el lenguaje de programación Java y el *framework* JMetal (Durillo and Nebro, 2011). Para resolver el problema de forma exacta se utilizó AMPL (Fourer et al., 1993) con la versión 12.6.3.0 de CPLEX (CPLEX Optimizer).

Se utilizó un servidor de cómputo del ClusterFING, la plataforma de computación de alto desempeño de la Universidad de la República (Nesmachnow, 2010). Las características del servidor utilizado son: servidor DELL Power Edge R720, dos procesadores Intel Xeon E52650 2,00GHz de 8 cores cada uno, 64GB de Memoria RAM, 600GB de almacenamiento en disco y sistema operativo Linux CentOS.

Para ejecutar CPLEX se utilizó un servidor con 5 TB de almacenamiento en disco, 16 GB de memoria RAM y un procesador Intel(R) Core(TM) i7 CPU975@3,33GHz de 4 cores.

6.4. Resultados de la configuración de parámetros del AE

En esta sección se presentan los resultados de la evaluación de parámetros del AE.

Para la métrica de HVR, las configuraciones con tamaño de población igual a 200 individuos ocuparon más veces las primeras cuatro posiciones de la clasificación. La Tabla 6.2 reporta que las configuraciones que ocuparon más veces las primeras cuatro posiciones de la clasificación tuvieron los valores $pC = 0,5$ y $pM = 0,001$.

Tabla 6.2: Cantidad de veces que cada configuración de parámetros ocupó las primeras cuatro posiciones de la clasificación de mejores resultados para hipervolumen relativo y tiempo de ejecución del AE, usando una población de 200 individuos.

Configuración		**Posición en HVR**				**Posición en tiempo**			
pM	pC	1º	2º	3º	4º	1º	2º	3º	4º
0,010	0,5	2	1	1	0	0	0	1	0
0,001	0,5	2	1	1	0	1	0	0	1
0,001	0,7	0	1	0	2	0	1	1	0
0,010	0,7	0	1	2	0	0	0	0	1

Utilizando la prueba de Kruskal-Wallis para los resultados obtenidos para la métrica HVR con distinto tamaño de población se obtuvieron p-valores menores a 5×10^{-2}, por lo tanto se puede asegurar la diferencia de las muestras con un nivel de confianza estadística mayor a 95 %. Al aplicar la prueba de K-S sobre las distribuciones de resultados de la métrica HVR, para configuraciones con distintos valores de pC y pM y tamaño de población fijo, los p-valores fueron superiores a 5×10^{-2}, entonces se utilizó el valor de las medianas del tiempo de ejecución del AE para cada configuración paramétrica como factor de desempate.

La mejor configuración de parámetros encontrada fue $\#P = 200$ y $pC = 0,5$ y $pM = 0,001$.

6.5. Resultados de la comparación del AE con las técnicas de referencia

En esta sección se reportan los resultados de la evaluación experimental y se comparan las soluciones del AE con las obtenidas por las otras técnicas empleadas para resolver el problema. Se presenta una comparación de soluciones obtenidas al resolver el problema de forma exacta, con el AE, con las técnicas greedys y con Round Robin.

6.5.1. Resultados para instancias en las que se puede hallar el óptimo de forma exacta

La Tabla 6.3 reporta la mediana del HVR, junto al RIQ, agrupado por instancias y por el método de resolución. Los valores mostrados para las instancias AMPL1, AMPL2 y AMPL3 indican que CPLEX es la técnica que tiene mejores valores de HVR (los más cercanos a 1). Para las tres instancias el AE tiene peores valores de HVR que la técnica CPLEX, pero la diferencia en el peor caso es apenas de 0,06. En dos de los casos CPLEX alcanza el valor ideal 1. En el caso de la instancia AMPL4 el AE obtiene un mejor valor de HVR mientras que CPLEX queda en segundo lugar con una diferencia de 0,13. Para todas las instancias la técnica greddy costo quedó en tercer lugar con un mínimo de diferencia de HVR con el segundo lugar de 0,20. Dados los valores de HVR que tienen el AE para el conjunto de cuatro instancias mencionado, se puede decir que el AE presenta una buena convergencia al frente de Pareto de referencia y al mismo tiempo sus soluciones tienen una buena diversidad. En todos los casos se cumple lo planteado en la Ecuación 6.3 para que exista significancia estadística en la diferencia entre la mediana del HVR del AE y

valor del HVR del resto de las técnicas.

Tabla 6.3: Hipervolumen relativo para las instancias del problema que se resolvieron de forma exacta. En el caso del AE se reporta el valor de la mediana y su rango intercuartílico.

Instancia	Técnica	HVR	RIQ
AMPL1	CPLEX	1,00	0,00
	AE	0,98	0,01
	Greedy costo	0,74	0,00
AMPL2	CPLEX	0,99	0,00
	AE	0,94	0,01
	Greedy costo	0,69	0,00
AMPL3	CPLEX	1,00	0,00
	AE	0,94	0,01
	Greedy costo	0,68	0,00
AMPL4	CPLEX	0,82	0,00
	AE	0,95	0,01
	Greedy costo	0,75	0,00

La Tabla 6.4 muestra, para todas las técnicas de resolución del problema, las mejores soluciones y las soluciones de compromiso para las cuatro instancias con las que se resolvió el problema de forma exacta. Se marca en negrita el mejor valor según costo o QoS.

En el caso de las mejores soluciones respecto a costo la Tabla 6.4 muestra que el AE para las instancias AMPL2, AMPL3 y AMPL4 el AE igualó el valor de CPLEX en el valor del costo y de QoS. Solo en el caso de la instancia AMPL1, el AE reportó un peor valor para el valor de costo respecto a CPLEX.

En el caso de las mejores soluciones respecto a QoS la Tabla 6.4 muestra que la técnica Greedy en QoS siempre obtuvo el mejor valor en la componente QoS. En el caso de la instancia AMPL1; el AE, CPLEX y la técnica Greedy QoS obtuvieron el mismo valor de QoS, pero si se comparan los valores de costo en estos tres casos el mejor valor lo obtuvo CPLEX, luego el AE y por último Greedy QoS. Para las instancias AMPL2 y AMPL4 el AE ocupa el segundo lugar en el valor de QoS, pero tiene mejor valor en la componente del costo de la solución que el reportado por la técnica Geedy QoS que ocupa el primer lugar.

Tabla 6.4: Resultado de cada técnica para los casos: solución de compromiso, mejor solución respecto a costo y mejor solución respecto a QoS.

		Solución de compromiso		Mejor solución en costo		Mejor solución en QoS	
Instancia	Técnica	Costo	QoS	Costo	QoS	Costo	QoS
AMPL1	CPLEX	0,41	3876,24	**0,13**	7380,25	0,98	**3036,11**
	AE	0,43	3922,67	0,15	7864,66	1,02	**3036,11**
	Greedy QoS	1,07	3036,11	1,07	3036,11	1,07	**3036,11**
	Greedy costo	0,51	5812,83	0,19	7380,25	0,67	5394,00
	Round Robin	1,64	10642,48	1,64	10642,48	1,64	10642,48
AMPL2	CPLEX	0,59	24504,13	**0,22**	45745,74	1,38	18048,38
	AE	0,61	29993,89	**0,22**	45745,74	2,00	17789,23
	Greedy QoS	2,35	17232,70	2,35	17232,70	2,35	**17232,70**
	Greedy costo	1,11	35634.35	0,44	45745,74	1,37	34226,50
	Round Robin	2,62	67629,22	2,62	67629,22	2,62	67629,22
AMPL3	CPLEX	0,59	20047,25	**0,22**	37782,11	1,50	13649,30
	AE	0,77	20372,47	**0,22**	37782,11	1,83	13970,86
	Greedy QoS	2,20	13474,58	2,20	13474,58	2,20	**13474,58**
	Greedy costo	0,95	29172,80	0,44	37782,11	1,08	28609,66
	Round Robin	2,64	56029,40	2,64	56029,40	2,64	56029,40
AMPL4	CPLEX	0,52	59557,83	**0,34**	83139,64	0,52	59557,83
	AE	1,04	51339,93	**0,34**	83139,64	2,77	32958,34
	Greedy QoS	2,98	31802,58	2,98	31802,58	2,98	**31802,58**
	Greedy costo	1,39	63223,03	0,45	83139,64	1,71	60829,69
	Round Robin	3,54	121857,86	3,54	121857,86	3,54	121857,86

La Tabla 6.5 reporta los tiempos de ejecución del AE en segundos. La máxima diferencia de tiempo se da entre las instancias AMPL1 y AMPL4, mientras que la mínima diferencia se da entre las instancias AMPL2 y AMPL3.

Tabla 6.5: Tiempo en segundos en el que el AE resolvió el problema para cada instancia.

Instancia	Promedio	Dev.est.	Mínimo	Máximo
AMPL1	119,71	10,61	108,23	130,96
AMPL2	158,39	13,33	144,09	171,96
AMPL3	156,66	15,01	140,80	173,62
AMPL4	227,98	17,20	210,18	246,81

En el caso de CPLEX los tiempos de ejecución se reportan en la Tabla 6.6. Las instancias con mayor diferencia de tiempos de ejecución entre sí son las instancias AMPL1 y AMPL4. Las instancia con mejor diferencia de tiempo entre sí son las instancias AMPL1 y AMPL2.

Tabla 6.6: Tiempo en segundos en el que CPLEX resolvió el problema para cada instancia.

Instancia	Tiempo
AMPL1	18414,79
AMPL2	30332,62
AMPL3	66421,85
AMPL4	120650,66

Los tiempos de ejecución del AE no se pueden comparar con los tiempos de ejecución de CPLEX porque se ejecutaron en plataformas distintas. Entre ambas técnicas coinciden las instancias con mayor diferencia en tiempo de ejecución. La proporción de la diferencia entre las instancias AMPL1 y AMPL4 en el caso de CPLEX es mucho mayor que en el caso del AE. Respecto a las instancias que presentan menos diferencia de tiempos de ejecución, en el caso del AE la diferencia proporcional entre los tiempos de estas instancias es mucho menor que en CPLEX.

Las heurísticas Greedy QoS, Greedy costo y Round Robin se ejecutaron en menos de un segundo para las 4 instancias analizadas. Por esta razón, dada la gran diferencia con el tiempo de ejecución de CPLEX y el AE, no se realiza un análisis detallado.

En la Figura 6.1 se presentan los frentes de Pareto generados para las instancias AMPL1, AMPL2, AMPL3 y AMPL4. En general, se observa que las heurísticas CPLEX, AE y Greedy QoS generan frentes de Paretos más diversos que las heurísticas Greedy Costo y Round Robin. Las heurísticas Greedy Costo y Round Robin generan frentes de Pareto muy poco diversos, con una tendencia a concentrarse en soluciones con valores de QoS o costo extremos.

El frente de Pareto de referencia para cada instancia está formado por soluciones de las técnicas CPLEX, AE y Greedy QoS. A excepción de la instancia AMPL4, una parte significativa del frente de Pareto de referencia está formada por soluciones de CPLEX, en segundo lugar el AE y por último la técnica Greddy QoS. En el caso particular de la instancia AMPL4 se puede apreciar que el AE logró cubrir la mayor diversidad de soluciones en la componente QoS y por esta razón sus soluciones formaron la mayor parte del frente de Pareto de referencia. Round Robin y Greedy costo no tienen soluciones que pertenezcan al frente de Pareto de referencia.

En términos generales se observa que a CPLEX le costó cubrir más soluciones en la componente QoS y en menor medida la técnica geedy costo también tuvo ese problema. En cuanto a Round Robin se puede decir que fue la peor técnica, solo obtuvo una solución no dominada y es la peor solución respecto a costo y QoS. El AE se caracterizó por tener una buena diversidad de soluciones y además sus soluciones, comparadas con las técnicas greedy y Round Robin, estuvieron más cercanas o directamente formaron al frente de Pareto de referencia.

La Tabla 6.7 muestra una comparación porcentual entre los valores de costo y QoS para las soluciones del AE y el resto de técnicas empleadas en la resolución del problema. En general, el AE muestra mejoras significativas respecto a las otras técnicas no exactas empleadas. Respecto a la técnica exacta CPLEX el AE presenta distintos comportamientos dependiendo del objetivo a optimizar.

Para las mejores soluciones respecto a QoS el AE mejora o iguala a la técnica CPLEX en 3 de las 4 instancias presentando una mejora máxima del 44,66 % en el caso de la instancia AMPL4. En el único caso en el que empeora los resultados respecto a la técnica

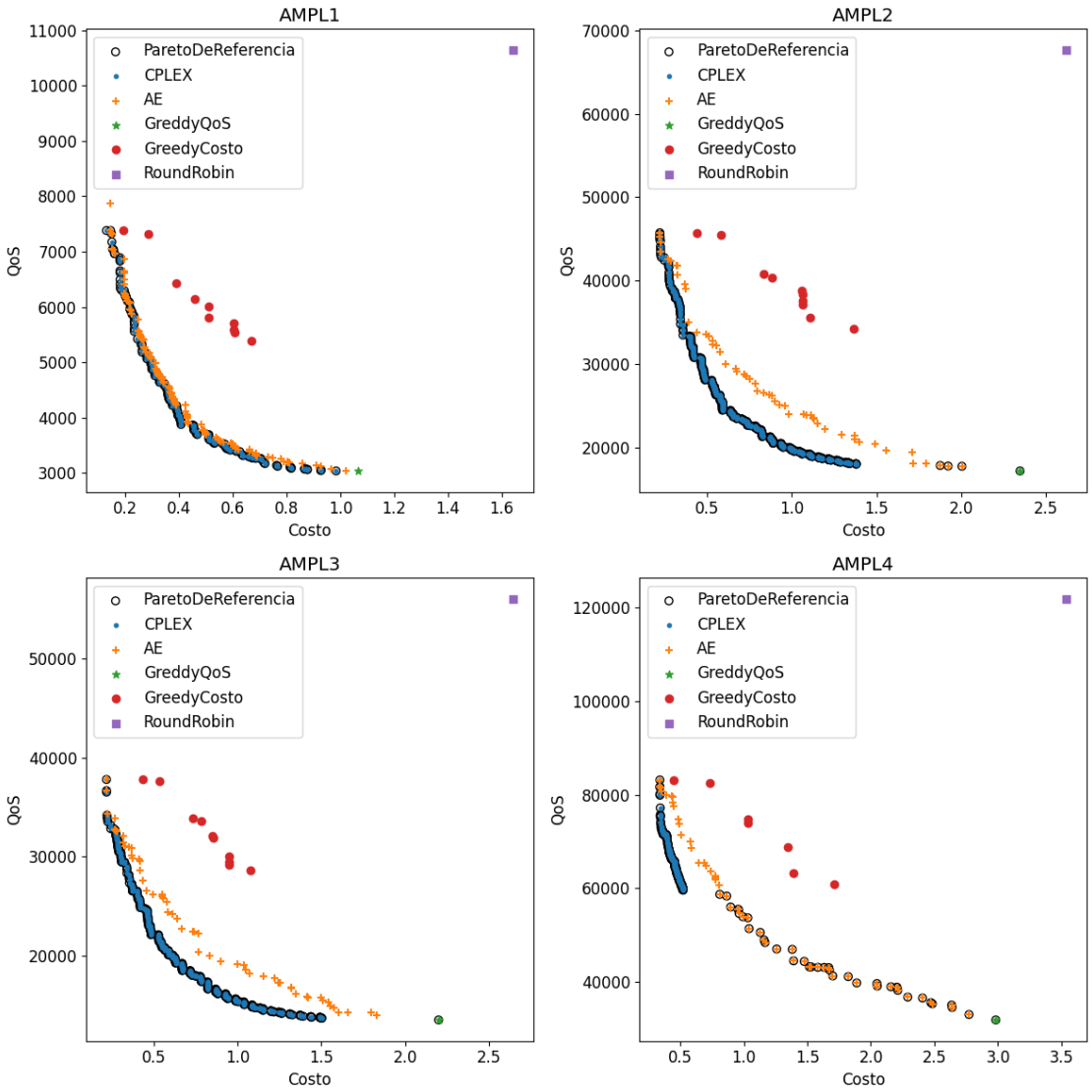

Figura 6.1: Frentes de Pareto para las instancias que se resolvieron de manera exacta.

CPLEX el porcentaje es pequeño 2,36 %. Respecto a la técnica Greedy QoS, que fue la que obtuvo los mejores resultados de QoS para las cuatro instancias, el AE igualó el resultado en una instancia y el peor porcentaje de desmejora fue solo de 3,68 %.

Respecto a las mejores soluciones en costo el AE iguala a la técnica CPLEX en 3 de las 4 instancias y solo empeora en un 15,38 % el resultado respecto a CPLEX en la instancia AMPL1. El AE mejora los resultados asociados el costo para todas las demás técnicas en todas las instancias presentando una mejora mínima de 21,05 % y una mejora máxima de 91,67 %.

En cuanto a las soluciones de compromiso, los resultados del AE no son tan buenos como en los casos de coso y QoS por separado. Sin embargo, el AE mejora costo y QoS a la vez de manera significativa para las técnicas Greedy costo y Round Robin para todas las instancias. Para el caso de greddy QoS, para todas las instancias, el AE mejora la componente de la solución asociada al costo y empeora en una proporción similar la

componente asociada a QoS. Al comparar con la técnica CPLEX el AE solo mejoró la componente de la solución QoS para una instancia, en el resto de los casos empeoró las soluciones.

En términos generales se concluye que el AE reportó buenos resultados en las mejores soluciones según QoS. La Figura 6.1 muestra que el AE tuvo una buena cobertura del frente de Pareto, lo que permitió explorar más soluciones con mejor valor de QoS que el resto de las técnicas empleadas. En cuanto a las mejores soluciones en costo, el AE tuvo mejoras significativas en términos generales debido a que sus soluciones presentaron coincidencias o gran proximidad a las soluciones exactas en esa zona del frente de Pareto. Por último, el AE tuvo peores soluciones de compromiso si se lo compara con la técnica CPLEX dado que el frente de Pareto del AE se posicionó más alejado del cero absoluto que el frente de Pareto de la técnica CPLEX. En este último caso se puede destacar que el AE tuvo mayor diversidad de soluciones que el resto de técnicas.

Tabla 6.7: Mejora porcentual de la técnica AE respecto a las técnicas CPLEX, Greedy costo, Greedy QoS y roun robin.

		Solución de compromiso		Mejor solución en costo	Mejor solución en QoS
Instancia	Heurística	Costo	QoS	Costo	QoS
AMPL1	CPLEX	-4,88 %	-1,20 %	-15,38 %	0,00 %
	Greedy QoS	59,81 %	-29,20 %	85,98 %	0,00 %
	Greedy costo	15,69 %	32,52 %	21,05 %	43,71 %
	Round Robin	73,78 %	63,14 %	90,85 %	71,47 %
AMPL2	CPLEX	-3,39 %	-22,40 %	0,00 %	1,44 %
	Greedy QoS	74,04 %	-74,05 %	90,64 %	-3,23 %
	Greedy costo	45,05 %	15,83 %	50,00 %	48,02 %
	Round Robin	76,72 %	55,65 %	91,60 %	73,70 %
AMPL3	CPLEX	-30,51 %	-1,62 %	0,00 %	-2,36 %
	Greedy QoS	65,00 %	-51,19 %	90,00 %	-3,68 %
	Greedy costo	18,95 %	30,17 %	50,00 %	51,17 %
	Round Robin	70,83 %	63,64 %	91,67 %	75,07 %
AMPL4	CPLEX	-100,00 %	13,80 %	0,00 %	44,66 %
	Greedy QoS	65,10 %	-61,43 %	88,59 %	-3,63 %
	Greedy costo	25,18 %	18,8 %	24,44 %	45,82 %
	Round Robin	70,62 %	57,87 %	90,40 %	72,95 %

6.5.2. Resultados para instancias en las que no se puede hallar el óptimo de forma exacta

La Tabla 6.8 reporta la mediana del HVR, junto al RIQ, agrupado por instancias y por el método de resolución. Debido a que las técnicas Greedy QoS y Round Robin generaron pocas soluciones no se pudro calcular el valor de HVR. Por lo tanto se reporta el valor de HVR para el AE y Greedy costo. Al comparar el valor del HVR entre las técnicas AE y Greedy costo para todas las instancias estudiadas en esta fase se cumplió que el AE

tuvo una mejora de aproximadamente el 30 % respecto a la técnica Greedy costo. Para estas instancias, al igual que en el caso donde se comparó el AE con la técnica exacta CPLEX, se puede concluir que el AE presenta una buena convergencia al frente de Pareto de referencia y al mismo tiempo sus soluciones tienen una buena diversidad. En todos los casos se cumple lo planteado en la Ecuación 6.3 para que exista significancia estadística en la diferencia entre la mediana del HVR del AE y valor del HVR de la técnica Greedy costo.

Tabla 6.8: Hipervolumen relativo para las instancias del problema que no se resolvieron de forma exacta. En el caso del AE se reporta el valor de la mediana y su rango intercuartílico.

Instancia	Técnica	HVR (mediana)	RIQ
CH1	AE	0,97	0,02
	Greedy costo	0,65	0,00
CH2	AE	0,98	0,01
	Greedy costo	0,67	0,00
M1	AE	0,97	0,01
	Greedy costo	0,64	0,00
M2	AE	0,99	0,01
	Greedy costo	0,67	0,00
G1	AE	0,98	0,01
	Greedy costo	0,58	0,00
G2	AE	0,99	0,01
	Greedy costo	0,59	0,00

La Tabla 6.9 muestra, para todas las técnicas de resolución del problema, las mejores soluciones y las soluciones de compromiso para las cuatro instancias con las que se resolvió el problema de forma exacta. Se marca en negrita el mejor valor según costo o QoS.

En el caso de las mejores soluciones respecto a costo la Tabla 6.4 muestra que el AE obtuvo el mejor resultado para todas las instancias.

En el caso de las mejores soluciones respecto a QoS la Tabla 6.4 muestra que la técnica Greedy en QoS obtuvo el mejor valor en la componente QoS para todas las instancias y que el AE para todas las instancias reportó el segundo mejor resultado para la componente de QoS. Un particularidad que se aprecia para la técnica Greedy en QoS es que para cada instancia se reporta para la técnica la misma solución para los casos de solución de compromiso, mejor solución en costo y mejor solución en QoS. El comportamiento descrito para la técnica Greedy en QoS se debe a que, para todas las instancias, la técnica generó soluciones con el mismo valor de la componente QoS y distintos valores en la componente de costo. Por lo tanto, las soluciones con el mismo valor de la componente QoS resultaron ser soluciones dominadas que se quitaron del conjunto de soluciones.

Tabla 6.9: Resultado cada heurística para los casos: solución de compromiso, mejor solución respecto a costo y mejor solución respecto a QoS. Resultados para las dos instancias chichas (CH1 Y CH2), las dos instancias medianas (M1 y M2) y las dos instancias grandes (G1 y G2)

		Solución de compromiso		Mejor solución en costo		Mejor solución en QoS	
Instancia	Heurística	Costo	QoS	Costo	QoS	Costo	QoS
CH1	AE	2,50	585984,97	**0,93**	1232226,27	4,20	466690,31
	Greedy QoS	8,24	459401,65	8,24	459401,65	8,24	**459401,65**
	Greedy costo	5,29	953486,60	3,70	1232226,27	5,77	906182,84
	Round Robin	11,58	1852399,54	11,58	1852399,54	11,58	1852399,54
CH2	AE	2,17	454303,20	**0,77**	984215,24	3,86	350421,55
	Greedy QoS	5,46	345948,19	5,46	345948,19	5,46	**345948,19**
	Greedy costo	2,92	772895,58	1,47	984215,24	3,26	745020,15
	Round Robin	7,88	1458439,78	7,88	1458439,78	7,88	1458439,78
M1	AE	3,42	924831,27	**1,32**	1911590,96	5,31	739137,93
	Greedy QoS	10,58	724694,41	10,58	724694,41	10,58	**724694,41**
	Greedy costo	6,98	1485437,59	4,86	1911590,96	7,34	1427574,96
	Round Robin	15,62	2891293,65	15,62	2891293,65	15,62	2891293,65
M2	AE	2,96	984155,87	**1,26**	1820476,88	5,47	682330,05
	Greedy QoS	8,04	681630,47	8,04	681630,47	8,04	**681630,47**
	Greedy costo	4,73	1418478,22	3,00	1820476,88	5,23	1364557,57
	Round Robin	10,90	2734637,54	10,90	2734637,54	10,90	2734637,54
G1	AE	4,11	1857605,50	**1,79**	4026547,82	6,34	1510223,57
	Greedy QoS	26,89	1491477,70	26,89	1491477,70	26,89	**1491477,70**
	Greedy costo	20,52	3150290,70	18,62	4026547,82	20,96	3041886,26
	Round Robin	29,75	6135782,68	29,75	6135782,68	29,75	6135782,68
G2	AE	3,45	1617467,97	**1,57**	3396253,84	5,67	1288672,94
	Greedy QoS	16,33	1270953,36	16,33	1270953,36	16,33	**1270953,36**
	Greedy costo	11,77	2634308,20	9,79	3396253,84	12,34	2554905,47
	Round Robin	20,13	5097385,12	20,13	5097385,12	20,13	5097385,12

La Tabla 6.9 muestra los tiempos de ejecución del AE para cada instancia utilizada en esta fase. Los tiempos son razonables para un AE que se ejecutará fuera de línea. En este contexto, el algoritmo no tiene que cumplir con restricciones de tiempo en tiempo real.

Las heurísticas Greedy QoS, Greedy costo y Round Robin se ejecutaron en menos de un segundo para las 6 instancias analizadas. Por esta razón, dada la gran diferencia con el tiempo de ejecución del AE, no se realiza un análisis detallado.

Tabla 6.10: Tiempo en segundos en el que el AE resolvió el problema para cada instancia.

Instancia	Promedio	Dev.est.	Mínimo	Máximo
CH1	4089,96	127,19	3881,54	4194,90
CH2	3620,40	156,12	3387,88	3798,56
M1	8282,22	237,05	8085,96	8882,42
M2	7985,92	292,59	7740,13	8485,45
G1	25103,32	510,92	24682,20	26432,02
G2	18928,89	564,76	18529,13	19985,84

En la Figura 6.2 se presentan los frentes de Pareto generados para las instancias CH1, CH2, M1, M2, G1 Y G2. Se puede apreciar como la técnica Round Robin genera una solución muy alejada del frente de Pareto de referencia. El frente de Pareto de referencia para cada instancia está formado por soluciones de las técnicas AE más la única solución no dominada de Greedy QoS. El AE fue la técnica con mayor diversidad de soluciones. La segunda técnica con mayor diversidad de soluciones fue la técnica Greedy costo pero como se aprecia en la figura las soluciones de la técnica se encuentran alejadas del frente de Pareto de referencia. En términos generales se observa que el AE genera soluciones más diversas y de mejor calidad que el resto de técnicas estudiadas en esta fase.

La Tabla 6.11 muestra una comparación porcentual entre los valores de costo y QoS para las soluciones del AE y el resto de técnicas empleadas en la resolución del problema en esta fase. En general, el AE muestra mejoras significativas respecto a las otras técnicas empleadas.

Para las mejores soluciones respecto a QoS el AE presenta una mejora mínima de 48,50 % y una mejora máxima de 75,97 %. Respecto a la técnica Greedy QoS, que fue la que obtuvo los mejores resultados de QoS para las seis instancias, el peor porcentaje de desmejora del AE fue solo de 1,99 %.

Respecto a las mejores soluciones en costo el AE mejora los resultados en todas las instancias y para todas las otras técnicas estudiadas. El AE presenta una mejora mínima del 47,62 % y una mejora máxima de 93,97 %. Cabe destacar que al comparar el AE con Greedy costo, que fue la segunda técnica con mejores soluciones en costo a nivel general para las seis instancias, el AE tiene un rango de mejoras que van desde 47,62 % hasta 90,39 %.

En cuanto a las soluciones de compromiso, el AE mejora costo y QoS a la vez de manera significativa para las técnicas Greedy costo y Round Robin para todas las instancias. En el caso de la técnica Greedy QoS para todas las instancias el AE solo mejora la componente costo de la solución y empeora la componente QoS.

A nivel general se puede concluir que el AE reportó buenos resultados al compararlo con el resto de técnicas empleadas en esta fase. A nivel de mejoras enfocadas en un solo objetivo del problema se destacan las mejoras a nivel de costo. Las mejoras a nivel de QoS también son significativas si se considera que en los casos que empeoró la solución lo hizo por un margen muy pequeño. A nivel de soluciones de compromiso se destaca también un buen resultado comparativo con las otras técnicas. El factor más significativo al momento de comparar las otras técnicas con el AE es que el AE presenta mucho mayor diversidad de soluciones que el resto de técnicas.

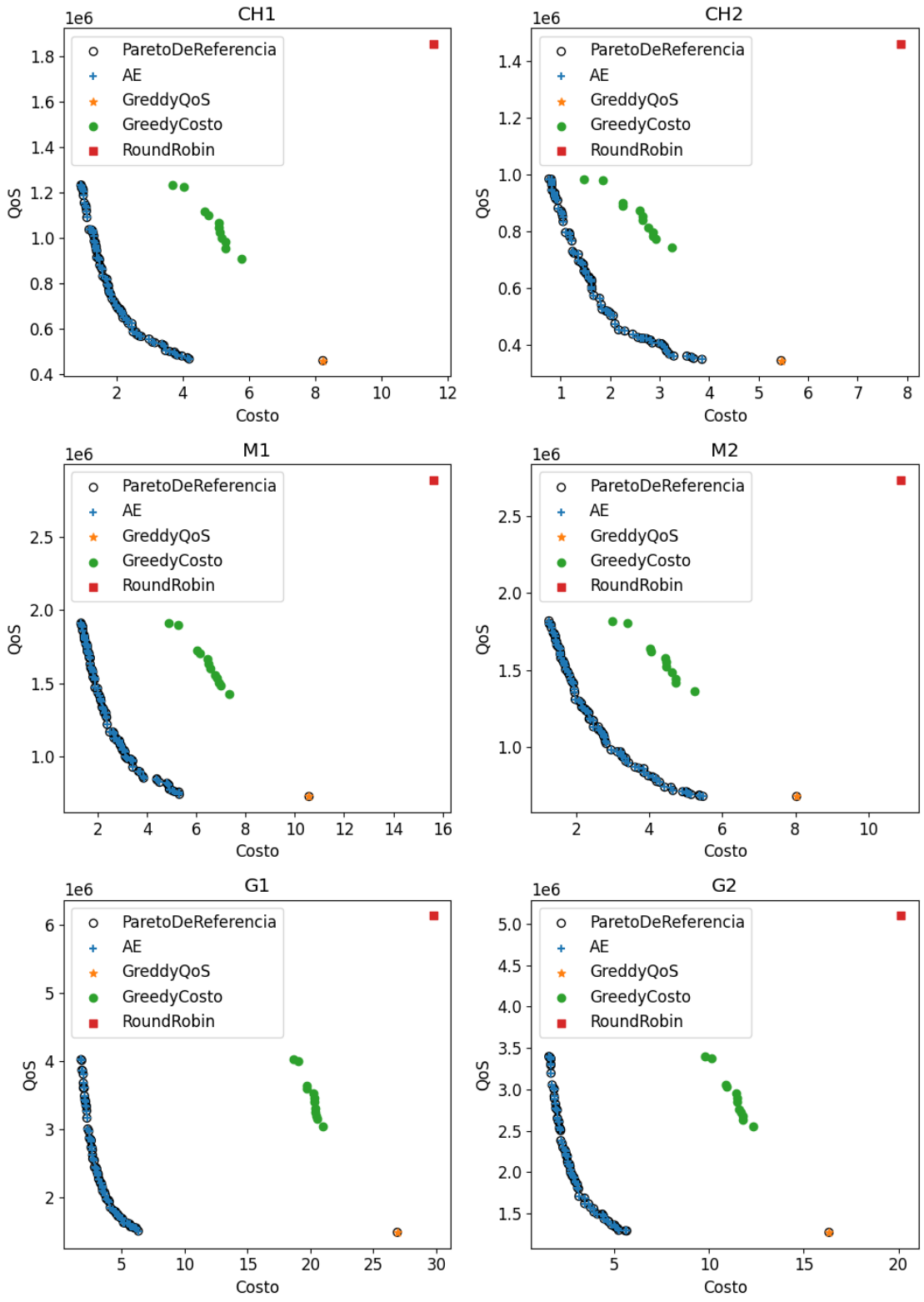

Figura 6.2: Frentes de Pareto para las instancias que no se resolvieron de manera exacta.

Tabla 6.11: Mejora porcentual de la técnica AE respecto a las técnicas Greedy costo, Greedy QoS y roun robin.

		Solución de compromiso		Mejor solución en costo	Mejor solución en QoS
Instancia	Heurística	Costo	QoS	Costo	QoS
CH1	Greedy QoS	69,66 %	-27,55 %	88,71 %	-1,59 %
	Greedy costo	52,74 %	38,54 %	74,86 %	48,50 %
	Round Robin	78,41 %	68,37 %	91,97 %	74,81 %
CH2	Greedy QoS	60,26 %	-31,32 %	85,90 %	-1,29 %
	Greedy costo	25,68 %	41,22 %	47,62 %	52,96 %
	Round Robin	72,46 %	68,85 %	90,23 %	75,97 %
M1	Greedy QoS	67,67 %	-27,62 %	87,52 %	-1,99 %
	Greedy costo	51,00 %	37,74 %	72,84 %	48,22 %
	Round Robin	78,10 %	68,01 %	91,55 %	74,44 %
M2	Greedy QoS	63,18 %	-44,38 %	84,33 %	-0,10 %
	Greedy costo	37,42 %	30,62 %	58,00 %	50,00 %
	Round Robin	72,84 %	64,01 %	88,44 %	75,05 %
G1	Greedy QoS	84,72 %	-24,55 %	93,34 %	-1,26 %
	Greedy costo	79,97 %	41,03 %	90,39 %	50,35 %
	Round Robin	86,18 %	69,73 %	93,98 %	75,39 %
G2	Greedy QoS	78,87 %	-27,26 %	90,39 %	-1,39 %
	Greedy costo	70,69 %	38,60 %	83,96 %	49,56 %
	Round Robin	82,86 %	68,27 %	92,20 %	74,72 %

Capítulo 7

Conclusiones y trabajo futuro

Este capítulo presenta las conclusiones del proyecto de grado sobre cómo construir redes de distribución de contenidos en la nube. También se detallan de manera concisa las principales líneas de trabajo futuro para mejorar los resultados actuales.

7.1. Conclusiones

Este proyecto de grado estudia el problema multiobjetivo que consiste en construir redes de distribución de contenidos en la nube. En el problema planteado se resuelven el aprovisionamiento de recursos de contenido en los centros de datos y el ruteo de los pedidos de los usuarios a dichos recursos de contenidos. Los objetivos de optimización son la minimización de los costos de máquinas virtuales, red y almacenamiento, y la maximización de la QoS para el usuario final.

El problema propuesto se divide en dos subproblemas: el aprovisionamiento de recursos en la nube y el enrutamiento de solicitudes de contenido. El algoritmo evolutivo diseñado resuelve los dos subproblemas en cada paso evolutivo hasta que alcanza buenos valores de las soluciones calculadas. Los resultados obtenidos muestran que el enfoque propuesto es adecuado para implementar CDN basadas en la nube a un costo reducido manteniendo altos valores de QoS.

En este proyecto de grado se incorporó un estimador de la QoS basado en mediciones de RTT lo cual mejora el enfoque adoptado en los trabajos relacionados basado en funciones de la distancia geográfica.

Publicaciones desarrolladas relacionadas con este proyecto de grado:

- Iturriaga, S., Goñi, G., Nesmachnow, S., Dorronsoro, B., Tchernykh, A. (2019). Cost and QoS Optimization of Cloud-Based Content Distribution Networks Using Evolutionary Algorithms. In: Meneses, E., Castro, H., Barrios Hernández, C., Ramos-Pollan, R. (eds) High Performance Computing. CARLA 2018. Communications in Computer and Information Science, vol 979. Springer, Cham. https://doi.org/10.1007/978-3-030-16205-4_22
- Iturriaga, S., Nesmachnow, S., Goñi, G. et al. Evolutionary Algorithms for Optimizing Cost and QoS on Cloud-based Content Distribution Networks. Program Comput Soft 45, 544–556 (2019). https://doi-org.proxy.timbo.org.uy/10.1134/S0361768819080127
- Goñi, G., Nesmachnow, S. Design of Content Distribution Networks for smart cities. VII Ibero-American Congress of Smart Cities. Enviado.

7.2. Trabajo futuro

Las principales líneas de trabajo futuro incluyen la construcción de un conjunto más grande y diverso de instancias del problema además de usar una función de QoS más precisa. Las instancias del problema se pueden mejorar incluyendo información histórica real de tráfico de recursos de contenidos en la red. La función de QoS se puede mejorar al incluir, además del RTT, el ancho de banda de la red. Un conjunto más preciso y amplio de instancias del problemas proporcionaría una visión más profunda sobre la efectividad del modelo propuesto. Por otro lado, una función QoS más precisa ayudaría a proporcionar soluciones más realistas.

El algoritmo evolutivo planteado es usado fuera de línea para resolver el problema, como trabajo futuro se puede mejorar la eficiencia el algoritmo para ejecutarlo en línea.

Bibliografía

J. Antunes. *Amazon AWS: Descomplicando a computação na nuvem.* Casa do Código, 2016. ISBN 9788555192388. URL https://books.google.com.uy/books?id=GXe7DQAAQBAJ.

Branch and Cut in CPLEX. Branch and cut in cplex. https://www.ibm.com/docs/en/icos/12.10.0?topic=concepts-branch-cut-in-cplex, 2022. Accedido: Abril 2022.

R. Buyya, J. Broberg, and A. Goscinski. *Cloud Computing: Principles and Paradigms.* Wiley Series on Parallel and Distributed Computing. Wiley, 2010. ISBN 9781118002209.

M. Chhetri, S. Chichin, V. Quoc Bao, and R. Kowalczyk. Smart cloud broker: Finding your home in the clouds. In *2013 IEEE/ACM 28th International Conference on Automated Software Engineering (ASE)*, pages 698 – 701, 2013. URL http://proxy.timbo.org.uy/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=edseee&AN=edseee.6693136&lang=es&site=eds-live.

Cisco. Cisco visual networking index: Global mobile data traffic forecast update, 2015 – 2020. Technical Report C11-738429-00, Cisco, 2016. URL https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.pdf.

CPLEX Optimizer. Cplex optimizer. https://www.ibm.com/analytics/cplex-optimizer, 2021. Accedido: Junio 2021.

R. E. Crandall, W. R. Crandall, and C. C. Chen. *Principles of supply chain management.* CRC press, 2009.

K. Deb. *Multi-objective optimization using evolutionary algorithms.* John Wiley & Sons, Chichester New York, 2001. ISBN 978-0471873396.

K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. Merelo, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature PPSN VI*, pages 849–858, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. ISBN 978-3-540-45356-7.

J. Durillo and A. Nebro. Jmetal: A java framework for multi-objective optimization. *Advances in Engineering Software*, 42(10):760–771, 2011. ISSN 09659978.

C. Ekstrom. *The R Primer, Second Edition.* CRC Press LLC Chapman and Hall/CRC, Place of publication not identified, 2017. ISBN 9781351803731.

I. Foster, Y. Zhao, I. Raicu, and S. Lu. Cloud computing and grid computing 360-degree compared. In *2008 Grid Computing Environments Workshop*, pages 1–10, 2008.

R. Fourer, D. Gay, and B. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Scientific Press, 1993. ISBN 9780894262333. URL https://books.google.com.uy/books?id=8vJQAAAAMAAJ.

R. Fourer, D. Gay, and B. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Scientific Press series. Thomson/Brooks/Cole, 2003. ISBN 9780534388096.

G. Gao, W. Zhang, Y. Wen, Z. Wang, and W. Zhu. Towards Cost-Efficient Video Transcoding in Media Cloud: Insights Learned from User Viewing Patterns. *IEEE Transactions on Multimedia*, 17(8):1286–1296, 2015. ISSN 15209210.

F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, 13(5):533 – 549, 1986. ISSN 0305-0548. doi: https://doi.org/10.1016/0305-0548(86)90048-1. Applications of Integer Programming.

F. Gravetter. *Statistics for the behavioral sciences*. Wadsworth, Belmont, CA, 2009. ISBN 0495602205.

M. Hu, J. Luo, Y. Wang, and B. Veeravalli. Practical resource provisioning and caching with dynamic resilience for cloud-based content distribution networks. *IEEE Transactions on Parallel and Distributed Systems*, 25(8):2169–2179, 2014. ISSN 10459219.

O. J. Jacobsen. Ripe atlas: A global internet measurement network. *The Internet Protocol Journal*, 18(3), sep 2015.

F. Jokhio, A. Ashraf, S. Lafond, and J. Lilius. A computation and storage trade-off strategy for cost-efficient video transcoding in the cloud. *Proceedings of the 39th Euromicro Conference Series on Software Engineering and Advanced Applications*, pages 365–372, 2013.

K. Jordahl, J. V. den Bossche, M. Fleischmann, J. Wasserman, J. McBride, J. Gerard, J. Tratner, M. Perry, A. G. Badaracco, C. Farmer, G. A. Hjelle, A. D. Snow, M. Cochran, S. Gillies, L. Culbertson, M. Bartos, N. Eubank, maxalbert, A. Bilogur, S. Rey, C. Ren, D. Arribas-Bel, L. Wasser, L. J. Wolf, M. Journois, J. Wilson, A. Greenhall, C. Holdgraf, Filipe, and F. Leblanc. geopandas/geopandas: v0.8.1, July 2020. URL https://doi.org/10.5281/zenodo.3946761.

K. V. Katsaros, G. Xylomenos, and G. C. Polyzos. Globetraff: A traffic workload generator for the performance evaluation of future internet architectures. In *2012 5th International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5, May 2012. doi: 10.1109/NTMS.2012.6208742.

Y. Liu, N. Chen, Z. Liu, and Y. Yang. Online cloud resource provisioning under cost budget for qos maximization. *2021 IEEE/ACM 29th International Symposium on Quality of Service (IWQOS), Quality of Service (IWQOS), 2021 IEEE/ACM 29th International Symposium on*, pages 1 – 6, 2021. ISSN 978-1-6654-1494-4. URL http://proxy.timbo.org.uy/login?url=https://search.ebscohost.com/login.aspx?direct=true&db=edseee&AN=edseee.9521347&lang=es&site=eds-live.

J. H. Maindonald. *Data analysis and graphics using R : an example-based approach.* Cambridge University Press, Cambridge New York, 2010. ISBN 9780521762939.

D. C. Marinescu. *Cloud computing: theory and practice.* Morgan Kaufmann, 2017.

M. Mattess, C. Vecchiola, S. K. Garg, and R. Buyya. Cloud bursting: Managing peak loads by leasing public cloud services. *Cloud Computing*, page 343–367, 2017. doi: 10.1201/b11149-16.

W. McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

S. Nesmachnow. Computación científica de alto desempeño en la facultad de ingeniería, universidad de la república. *Revista de la Asociación de Ingenieros del Uruguay*, 61: 12–15, 01 2010.

S. Nesmachnow, S. Iturriaga, and B. Dorronsoro. Efficient heuristics for profit optimization of virtual cloud brokers. *IEEE Computational Intelligence Magazine*, 10(1):33–43, 2015.

OpenFING. Openfing. `https://open.fing.edu.uy/`, 2017. Accedido: Setiembre 2017.

K. Park and W. Willinger. *Sele-Similar network traffic and performance evaluation.* Wiley & Son, 2000.

S. Seabold and J. Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

C. Spatz. *Basic statistics : tales of distributions.* Wadsworth, Australia, 2010. ISBN 9780495808916.

N. Srinivas and K. Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994. doi: 10.1162/evco.1994.2.3.221.

L. Wolsey. *Integer Programming.* Wiley, 2020. ISBN 9781119606536.

W. Xiao, W. Bao, X. Zhu, C. Wang, L. Chen, and L. T. Yang. Dynamic request redirection and resource provisioning for cloud-based video services under heterogeneous environment. *IEEE Transactions on Parallel and Distributed Systems*, 27(7):1954–1967, 2016. ISSN 10459219.

J. Zhang, H. Huang, and X. Wang. Resource provision algorithms in cloud computing: A survey. *Journal of Network and Computer Applications*, 64:23–42, 2016. ISSN 10958592.