Universidad Nacional de Asunción
Facultad Politécnica

Ingeniería en Informática

Scalable and Decentralized Urban Traffic Optimization with Verifiable Storage for Smart City Deployments

Kevin Mathias Galeano Saldivar
María José Duarte Kowalewski

Proyecto de Trabajo de Grado presentado en conformidad a los requisitos para obtener el grado de Ingeniería en Informática
San Lorenzo - Paraguay
Febrero - 2026

Universidad Nacional de Asunción
Facultad Politécnica

Ingeniería en Informática

Scalable and Decentralized Urban Traffic Optimization with Verifiable Storage for Smart City Deployments

Kevin Mathias Galeano Saldivar
María José Duarte Kowalewski

San Lorenzo - Paraguay
Febrero - 2026

Universidad Nacional de Asunción
Facultad Politécnica

Ingeniería en Informática

Scalable and Decentralized Urban Traffic Optimization with Verifiable Storage for Smart City Deployments

Kevin Mathias Galeano Saldivar
María José Duarte Kowalewski

Asesor:
PhD Marcos Villagra

Proyecto de Trabajo de Grado presentado en conformidad a los
requisitos para obtener el grado de Ingeniería en Informática
San Lorenzo - Paraguay
Febrero - 2026

Dedico a mi blah blah blah,
a mi blah blah blah.

AGRADECIMIENTOS

Titulo1
Titulo2

Autor: Nombre y apellido

Asesor: Nombre y apellido

RESUMEN

Este es el resumen

Palabras Clave:

Title

Author: full name

Advisor: full name

ABSTRACT

This is the abstract

Keywords:

ÍNDICE

Página

LISTA DE FIGURAS

Página

LISTA DE TABLAS

Página

LISTA DE SÍMBOLOS

CAPÍTULO 1

INTRODUCCIÓN

1.1. Introducción

CAPÍTULO 2
MARCO TEÓRICO

2.1. Blockchain

Blockchain es una tecnología descentralizada que utiliza una cadena de bloques enlazados criptográficamente para registrar transacciones de manera inmutable y distribuida. Esta estructura básica permite la verificación colectiva sin intermediarios confiables, estableciendo las bases para sistemas de consenso distribuido que garantizan la integridad de los datos mediante mecanismos criptográficos y protocolos de validación colectiva.

Fundamentos de Blockchain

Un blockchain consiste en una secuencia de bloques, donde cada bloque contiene un conjunto de transacciones válidas, una marca de tiempo, un número usado una única vez (nonce) y el hash criptográfico del bloque anterior, lo que garantiza la inmutabilidad mediante la dependencia unidireccional de los hashes. Esta estructura hace que cualquier modificación en un bloque propague cambios en todos los subsiguientes, detectándose fácilmente por la red distribuida [1, 2].

La arquitectura básica opera en una red peer-to-peer donde nodos mantienen copias completas del ledger, validando transacciones mediante funciones hash como SHA-256 para enlazar bloques y prevenir alteraciones. La inmutabilidad surge de la dependencia unidireccional de los hashes, haciendo el ledger resistente a manipulaciones retrospectivas una vez que un bloque alcanza suficiente profundidad en la cadena [3, 4]. Esta característica fundamental permite que blockchain funcione como un registro distribuido e inmutable, donde la confianza se establece mediante consenso algorítmico en lugar de autoridades centralizadas.

La Figura 2.1 ilustra la estructura básica de un bloque típico en una cadena blockchain, mostrando cómo cada bloque referencia criptográficamente al bloque anterior mediante su hash,

creando una cadena inmutable.

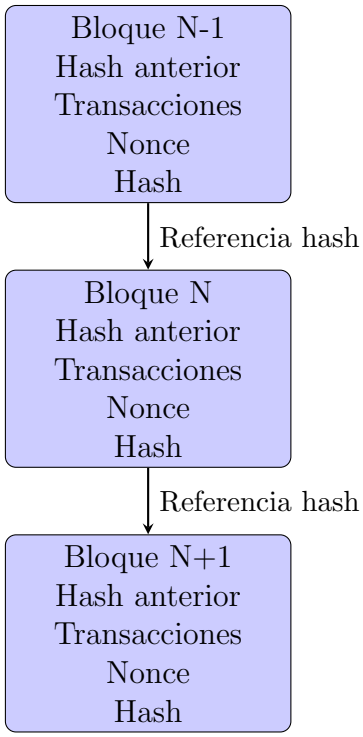

Figura 2.1: Estructura básica de una cadena de bloques, mostrando la dependencia criptográfica entre bloques consecutivos.

Arquitectura de Cadena Lineal

Bitcoin y Ethereum ejemplifican arquitecturas de cadena lineal (o unidireccional), donde bloques se apendan secuencialmente formando una estructura cronológica irreversible. En Bitcoin, la cadena principal emerge del fork con mayor trabajo computacional acumulado (regla de la cadena más larga), mientras que Ethereum mantiene compatibilidad similar pero soporta estado mutable mediante contratos inteligentes ejecutables en una máquina virtual descentralizada [1, 5].

Esta linealidad asegura orden total en transacciones, diferenciándose de estructuras DAG (Directed Acyclic Graph) no lineales que permiten validación paralela. Bitcoin se enfoca principalmente en transferencias de valor, mientras que Ethereum extiende esta funcionalidad a contratos inteligentes y aplicaciones descentralizadas, ampliando significativamente las capacidades de la tecnología blockchain [5, 2].

La arquitectura lineal presenta ventajas en términos de simplicidad y orden determinístico, pero introduce limitaciones en escalabilidad y throughput, ya que todas las transacciones deben procesarse secuencialmente en la cadena principal.

Mecanismos de Consenso

El consenso en blockchain resuelve el problema de acuerdo distribuido en entornos sin confianza, garantizando que todos los nodos honestos lleguen a un acuerdo sobre el estado del

ledger. Los mecanismos de consenso más ampliamente adoptados son Proof-of-Work (PoW) y Proof-of-Stake (PoS), cada uno con características distintivas que equilibran seguridad, descentralización y eficiencia energética.

Proof-of-Work (PoW) requiere que los mineros compitan resolviendo puzzles computacionales complejos para proponer bloques válidos, asegurando que la cadena más larga represente la verdad aceptada por la mayoría de la potencia computacional de la red. Este mecanismo, implementado originalmente en Bitcoin, proporciona seguridad mediante el costo computacional requerido para modificar el historial, pero consume cantidades significativas de energía [1].

Proof-of-Stake (PoS) selecciona propositores de bloques proporcionalmente a su participación económica (stake) en la moneda, reduciendo consumo energético al reemplazar trabajo computacional por compromiso económico. Ethereum adoptó PoS en su transición post-merge, logrando una reducción estimada del 99.9 % en consumo energético mientras mantiene propiedades de seguridad [5].

Ambos mecanismos conceptualmente priorizan liveness (progreso continuo de la cadena) y safety (consistencia del estado), tolerando hasta cierto umbral de nodos adversarios corruptos según el modelo de seguridad asumido. La Tabla 2.1 sintetiza las principales diferencias entre ambos mecanismos.

Cuadro 2.1: Comparación entre Proof-of-Work y Proof-of-Stake

Criterio	**Proof-of-Work (PoW)**	**Proof-of-Stake (PoS)**
Consumo energético	Alto (minería intensiva)	Bajo (validación ligera)
Velocidad de confirmación	Lenta (10 min Bitcoin)	Rápida (12 s Ethereum PoS)
Requisitos de hardware	Especializado (ASIC, GPU)	Estándar (servidores)
Seguridad	Basada en potencia computacional	Basada en stake económico
Descentralización	Alta (acceso abierto a minería)	Variable (requiere capital inicial)
Resistencia a ataques	51 % de hash rate	51 % de stake económico
Ejemplos	Bitcoin, Ethereum pre-merge	Ethereum post-merge, Cardano

Limitaciones Principales

Las blockchains lineales enfrentan limitaciones fundamentales que restringen su aplicabilidad en escenarios que requieren alto throughput, baja latencia o costos transaccionales reducidos. El trilema de escalabilidad (seguridad, descentralización, escalabilidad) establece que es difícil optimizar simultáneamente los tres aspectos, resultando en trade-offs inherentes [3].

El throughput limitado representa una restricción crítica: Bitcoin procesa aproximadamente 7 transacciones por segundo (TPS), mientras que Ethereum alcanza 20-30 TPS en condiciones normales, valores insuficientes para aplicaciones de alta frecuencia como sistemas de monitoreo urbano que requieren escritura frecuente de datos [6]. La latencia también constituye un desafío, con tiempos de bloque de 10 minutos en Bitcoin y 12 segundos en Ethereum, lo cual puede resultar inadecuado para aplicaciones que requieren confirmaciones rápidas.

Los costos transaccionales presentan variabilidad significativa, especialmente durante períodos de congestión de red. En Ethereum, los fees pueden oscilar entre $0.50 y más de $10 USD por transacción dependiendo del estado de la red, lo cual resulta prohibitivo para aplicaciones que requieren escritura frecuente de datos, como el reporte periódico de métricas de tráfico [7, 8].

La escalabilidad adicional se ve limitada por el crecimiento continuo del tamaño del ledger, que exige nodos completos con alto almacenamiento y ancho de banda, dificultando la participación de dispositivos con recursos limitados. Soluciones como sharding o capas secundarias (layer-2) mitigan parcialmente estos problemas pero introducen trade-offs en descentralización y complejidad operativa [6, 3].

Estas limitaciones motivan la exploración de arquitecturas alternativas, como BlockDAG, que ofrecen mejoras significativas en throughput, latencia y costos mediante estructuras no lineales que permiten procesamiento paralelo de transacciones.

2.2. BlockDAG

BlockDAG (Block Directed Acyclic Graph) representa una generalización arquitectónica de la estructura blockchain tradicional, donde los bloques no referencian a un único padre, sino a múltiples padres denominados tips o puntas del grafo [9]. Esta arquitectura permite superar limitaciones fundamentales de las blockchains lineales mediante la incorporación de bloques paralelos válidos, transicionando de un modelo de exclusión competitiva a un modelo de inclusión colaborativa [10].

Concepto de DAG vs. Cadena Lineal

La diferencia fundamental entre blockchain y BlockDAG reside en la topología de la estructura de datos subyacente. En una blockchain tradicional, cada bloque B_i mantiene una referencia criptográfica exactamente a un bloque padre B_{i-1}, formando una estructura de cadena lineal donde únicamente una rama es válida según reglas de consenso como la cadena más larga o más pesada. Las ramas paralelas generadas por colisiones de minería se descartan como bloques huérfanos (orphans), desperdiciando el trabajo computacional invertido en su creación.

En contraste, un BlockDAG permite que cada bloque B_i referencie a un conjunto de padres $P \subset \{B_0, ..., B_{i-1}\}$, formando matemáticamente un grafo dirigido acíclico $G = (V, E)$ donde V representa el conjunto de bloques y E representa las referencias hash que apuntan a bloques anteriores. La propiedad de aciclicidad garantiza que no existen caminos que permitan regresar a un bloque previo, preservando la integridad temporal, mientras que la estructura de grafo permite bifurcaciones válidas concurrentes que se integran en la topología general [9].

Esta transición arquitectónica modifica el paradigma operativo: de un modelo de exclusión donde los mineros compiten para generar el siguiente bloque único, a un modelo de inclusión donde múltiples bloques generados simultáneamente son válidos y colaboran para extender el grafo en paralelo, maximizando la utilización del ancho de banda de la red [10].

Ventajas Estructurales: Paralelismo, Throughput y Latencia

El paralelismo inherente en BlockDAG elimina restricciones artificiales de throughput presentes en blockchains lineales. En sistemas como Bitcoin, el intervalo entre bloques debe ser suficientemente grande para asegurar la propagación completa del bloque anterior a través de la red antes de que se genere el siguiente, evitando bifurcaciones masivas que comprometan la seguridad. Esta restricción limita el throughput a valores como 7 transacciones por segundo, independientemente del ancho de banda disponible.

BlockDAG permite que múltiples bloques generados simultáneamente sean válidos e integrados en el grafo, rompiendo la restricción de ün bloque a la vezz permitiendo utilizar todo el ancho de banda de la red para procesar transacciones en paralelo [9]. Esta capacidad de procesamiento paralelo se traduce en throughput significativamente mayor, limitado principalmente por el ancho de banda físico de la red en lugar de restricciones algorítmicas.

La latencia también se beneficia de esta arquitectura. En blockchains tradicionales, la seguridad se basa parcialmente en la probabilidad de orfandad: bloques generados en paralelo compiten, y solo uno prevalece mientras los demás se descartan. Esta competencia requiere tiempos de bloque largos (10 minutos en Bitcoin) para mantener baja la probabilidad de colisiones. En BlockDAG, como los bloques paralelos no se descartan sino que se integran, los mineros no enfrentan el riesgo de perder recompensas por colisiones, permitiendo tiempos de bloque mucho más cortos, incluso sub-segundo en implementaciones modernas como Kaspa [10].

Adicionalmente, BlockDAG permite desacoplar el throughput de la latencia, mitigando parcialmente el denominado "blockchain trilemma"que establece la dificultad de maximizar simultáneamente seguridad, descentralización y escalabilidad en sistemas distribuidos [11]. En protocolos tipo Nakamoto, aumentar la tasa o tamaño de bloques suele implicar reducir seguridad o descentralización, ya que incrementa la probabilidad de bloques huérfanos y eleva los requisitos de hardware para nodos completos [12]. En contraste, los protocolos DAG-based permiten procesar transacciones en paralelo, mejorando throughput y reduciendo latencia de confirmación respecto a blockchains lineales [12]. Esta capacidad de incorporar bloques paralelos en la estructura de consenso permite aumentar la tasa de bloques y el throughput sin degradar la seguridad al mismo ritmo que en protocolos tipo Nakamoto [10], ya que los bloques paralelos se integran como parte del conjunto de bloques honestos bien conectados y contribuyen a la seguridad acumulativa del sistema, en lugar de ser descartados como en una cadena lineal.

Consenso GHOSTDAG/PHANTOM: Fundamentos Conceptuales

Un DAG define naturalmente un orden parcial entre bloques: se sabe que un bloque A precedió a un bloque B si existe un camino dirigido de A a B en el grafo. Sin embargo, las transacciones financieras requieren un orden total para resolver conflictos como el doble gasto: es necesario determinar de manera unívoca qué transacción ocurrió primero cuando dos

transacciones intentan gastar el mismo activo.

PHANTOM representa el protocolo teórico que aborda este desafío mediante la distinción entre bloques creados por nodos honestos y atacantes basándose en la conectividad del grafo. El protocolo busca identificar el k-cluster más grande, definido como un subconjunto de bloques bien conectados que representan la actividad honesta de la red. Sin embargo, encontrar el k-cluster máximo es un problema computacionalmente intratable (NP-hard), lo que limita su implementación práctica directa [10].

GHOSTDAG constituye una variante práctica y eficiente de PHANTOM, implementando una aproximación "greedy"(voraz) que resuelve el problema de ordenamiento de manera computacionalmente viable. El protocolo utiliza un algoritmo de coloración que clasifica los bloques en dos conjuntos: bloques Azules, que representan bloques honestos bien conectados y generados en tiempo, y bloques Rojos, que representan bloques potencialmente atacantes, retardados o desconectados del grafo principal.

La cadena azul se construye seleccionando la "mejor punta"del grafo basándose en el mayor peso acumulado (similar al concepto de cadena más larga en Bitcoin) y recorriendo hacia atrás para formar una cadena principal que representa el consenso honesto. Una vez definida la cadena azul, todos los bloques del DAG, incluidos los rojos y aquellos fuera de la cadena principal, se ordenan topológicamente, creando un orden canónico único que todos los nodos respetan para resolver conflictos de transacciones [10].

A diferencia de Bitcoin, que descarta completamente las ramas paralelas, GHOSTDAG las penaliza en el ordenamiento pero preserva su información (data availability), invalidando únicamente transacciones conflictivas dentro de bloques rojos mientras mantiene la contribución de seguridad de todos los bloques integrados en el grafo.

La Tabla 2.2 sintetiza las principales diferencias entre arquitecturas blockchain tradicionales y BlockDAG basadas en GHOSTDAG. La comparación considera protocolos representativos de cada paradigma: Nakamoto consensus implementado en Bitcoin y Ethereum para blockchains lineales [1, 5, 13, 14], y PHANTOM/GHOSTDAG como generalización teórica y práctica del consenso distribuido en estructuras DAG [10]. Los valores de throughput y latencia reflejan métricas observadas en redes operativas, donde Bitcoin mantiene un intervalo de bloque de 10 minutos para asegurar propagación completa, Ethereum PoS opera con bloques de 12 segundos, e implementaciones BlockDAG como Kaspa alcanzan latencias sub-segundo con throughput significativamente mayor [15, 16].

Estas ventajas estructurales hacen de BlockDAG una arquitectura prometedora para aplicaciones que requieren alto throughput, baja latencia y costos transaccionales reducidos, como sistemas de monitoreo urbano que necesitan escritura frecuente de datos con verificación descentralizada.

Cuadro 2.2: Comparación entre Blockchain Tradicional y BlockDAG

Característica	**Blockchain Tradicional**	**BlockDAG (GHOSTDAG)**
Topología	Lineal (1 padre)[10]	Grafo acíclico (múltiples padres)[10, 16]
Bloques huérfanos	Descartados (seguridad desperdiciada)[10]	Integrados como tíos/padres (seguridad acumulativa)
Ordenamiento	Total (intrínseco a la cadena)	Parcial (topológico) → Total (vía algoritmo)[1
Escalabilidad	Limitada por latencia de propagación[10]	Alta (limitada por ancho de banda físico)[10, 1
Confirmación	Probabilística lenta (e.g., 6 bloques)	Probabilística rápida (segundos)[16, 15]
Minería	Competitiva (winner-takes-all)[10]	Cooperativa (pay-per-share implícito)[10]
Throughput típico	7-30 TPS[1, 5]	100+ TPS (depende de implementación)[15, 1
Latencia de bloque	10 min (Bitcoin)[13], 12 s (Ethereum)[14]	Sub-segundo a segundos[15, 16]

2.3. IPFS

El InterPlanetary File System (IPFS) es un protocolo peer-to-peer de sistema de archivos distribuido diseñado para conectar todos los dispositivos de cómputo con el mismo sistema de archivos, utilizando direccionamiento basado en contenido (content-addressing) en lugar de ubicación (location-addressing como HTTP) [17]. Esta arquitectura fundamental permite la distribución descentralizada de datos, la deduplicación automática y la verificación criptográfica de integridad, posicionando a IPFS como una solución complementaria a blockchains y BlockDAG para el almacenamiento de grandes volúmenes de datos con trazabilidad verificable.

Arquitectura IPFS: Sistema de Archivos Distribuido

IPFS está estructurado como una pila de sub-protocolos modulares e intercambiables, organizados en siete capas funcionales que proporcionan identidad, enrutamiento, intercambio de bloques y versionado de contenido [17]. La capa de Identidades genera y verifica la identidad de nodos mediante NodeId = hash(PublicKey), siguiendo el esquema de identidad basado en PKI descrito en la especificación de arquitectura IPFS [17, 18]. Este esquema utiliza criptografía S/Kademlia para resistencia a ataques Sybil, donde los nodos son identificados por un NodeId que corresponde al hash criptográfico de una clave pública, creado mediante el puzzle criptográfico estático de S/Kademlia [17]. La capa de Red soporta múltiples protocolos de transporte (WebRTC, uTP/LEDBAT) con traversal de NAT mediante ICE y direccionamiento multiaddr agnóstico a la capa de transporte.

La capa de Routing implementa una Tabla de Hash Distribuida (DHT) basada en S/Kademlia y Coral DSHT [**freedman2004coral**, **blockchainMeetsDFS**, 17]. Coral DSHT extiende Kademlia de tres formas particularmente importantes: organiza una jerarquía de DSHTs separadas llamadas clusters dependiendo de la región y tamaño, optimiza la latencia mediante una infraestructura de indexación jerárquica, y utiliza una abstracción denominada Distributed Sloppy Hash Table (DSHT) [**freedman2004coral**, 17]. IPFS logra esto utilizando una DSHT basada en S/Kademlia y Coral, almacenando valores pequeños directamente y valores grandes como referencias (provider records) [**blockchainMeetsDFS**, 17]. El protocolo de Intercambio

(BitSwap) permite que los nodos intercambien bloques mediante un sistema de crédito/débito y estrategias de envío probabilísticas que incentivan la distribución de contenido y resisten comportamientos de nodos que solo consumen sin compartir, inspirado conceptualmente en BitTorrent.

La capa de Objetos implementa un Merkle DAG (Grafo Acíclico Dirigido) donde cada objeto contiene links (IPFSLink) y data (byte array), donde los enlaces son hashes criptográficos (multihash) de los objetos destino. La capa de Archivos proporciona un modelo de objetos versionado similar a Git, con tipos blob (datos brutos), list (secuencias de bloques para deduplicación), tree (directorios) y commit (snapshots de historial). Finalmente, la capa de Nombres (IPNS) ofrece un sistema de nombres mutable auto-certificado usando criptografía de clave pública para mapear NodeId a CIDs mutables.

Filosóficamente, IPFS puede visualizarse como un único swarm de BitTorrent intercambiando objetos dentro de un solo repositorio Git"[17], combinando DHT (Kademlia), intercambio de bloques (BitTorrent), versionado (Git) y auto-certificación (SFS). Esta arquitectura proporciona propiedades clave: content addressing (cada objeto identificado por su checksum multihash), resistencia a alteraciones (verificación mediante checksum), y deduplicación automática (objetos con contenido idéntico comparten el mismo hash y se almacenan una sola vez).

La Figura 2.2 ilustra la arquitectura de pila de IPFS con sus siete capas funcionales, mostrando la organización jerárquica desde la capa de aplicación hasta la capa de red.

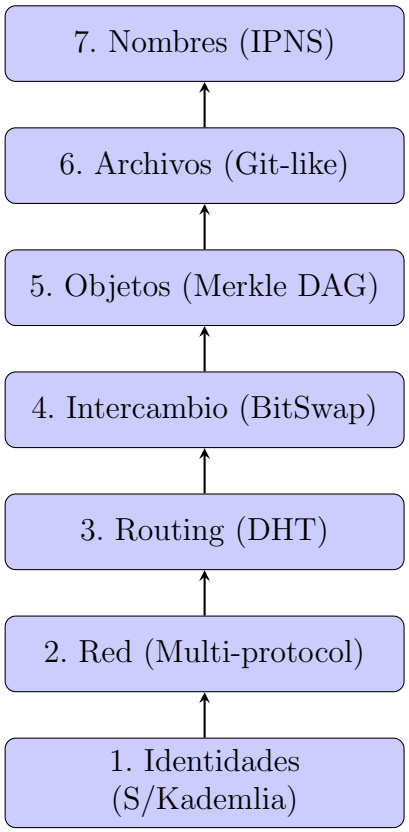

Figura 2.2: Arquitectura de pila de IPFS con sus siete capas funcionales.

Content Identifiers (CIDs): Estructura y Hash Criptográfico

Un Content Identifier (CID) es un identificador auto-descriptivo que contiene tanto el codec (formato de interpretación de datos) como un multihash (hash auto-descriptivo que especifica la función hash utilizada) [17]. La estructura de multihash sigue el formato $\langle function\_code \rangle \langle digest\_length \rangle \langle$ donde function_code identifica el algoritmo hash (ej. SHA-256, BLAKE3), digest_length especifica la longitud del hash en bytes, y digest_bytes contiene el hash criptográfico en sí.

IPFS soporta dos versiones de CID. CIDv0 (legacy) utiliza un formato simplificado donde el multihash se codifica en Base58, produciendo identificadores de 46 caracteres que comienzan con "Qm". CIDv1 (moderno) emplea un formato extensible con prefijos explícitos: $\langle multibase\_prefix \rangle \langle cid\_version \rangle \langle multicodec \rangle \langle multihash \rangle$, donde multibase_prefix indica la codificación base utilizada (ej. base32, base58btc), cid_version especifica la versión del CID, multicodec identifica el formato del contenido objetivo (dag-pb/UnixFS, dag-cbor, raw), y multihash contiene el hash del contenido.

Las propiedades fundamentales de los CIDs incluyen determinismo (el mismo contenido produce el mismo CID en cualquier nodo), inmutabilidad (cualquier modificación genera un CID diferente), y forward-compatibility (CIDv1 permite evolucionar algoritmos hash y codecs sin romper compatibilidad). Para archivos grandes, IPFS los divide en chunks, donde el CID corresponde al hash del bloque raíz del DAG, no del archivo completo, permitiendo chunking flexible, deduplicación y Merkle trees balanceados. Los CIDs se utilizan en URIs de la forma /ipfs/⟨CID⟩/⟨path⟩, donde IPFS resuelve el path recorriendo el Merkle DAG desde el hash raíz y navegando por links nombrados.

La Figura 2.3 muestra el flujo de resolución de un CID, desde la petición del usuario hasta la reconstrucción del contenido mediante DHT y BitSwap.

Pinning y Gateways: Persistencia y Acceso

Los nodos IPFS cachean contenido descargado en almacenamiento local, pero este es finito. Un proceso de garbage collection periódico elimina bloques no referenciados para liberar espacio. El pinning (anclaje) marca objetos y sus bloques para que no sean eliminados durante la garbage collection, garantizando persistencia local indefinida [17]. El pinning recursivo ancla un objeto raíz y todos sus descendientes recursivamente, preservando estructuras completas como árboles de archivos y DAGs versionados.

Los servicios de pinning (pinning services) son terceros que ejecutan nodos IPFS dedicados para anclar datos de usuarios a cambio de una tarifa, asegurando alta disponibilidad y redundancia geográfica. Esta infraestructura es crítica para aplicaciones que requieren garantías de persistencia sin mantener infraestructura propia. Sin embargo, el pinning requiere que al menos un nodo mantenga el contenido vivo; si todos los nodos eliminan el contenido (unpinning), este puede volverse irrecuperable.

Los IPFS Gateways son servidores HTTP que exponen contenido IPFS a navegadores web tradicionales sin requerir que el usuario ejecute un nodo IPFS completo. El funcionamiento

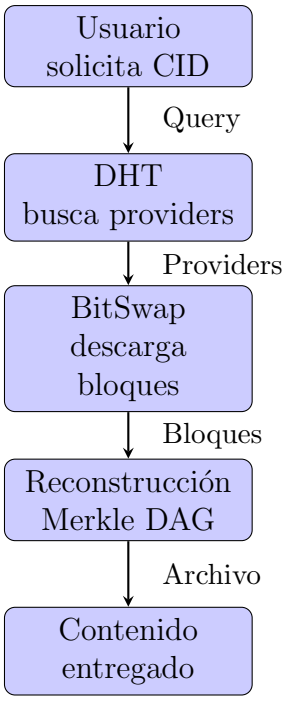

Figura 2.3: Flujo de resolución de CID en IPFS: desde la solicitud del usuario hasta la entrega del contenido.

típico implica: recepción de petición HTTP en la forma https://⟨gateway_url⟩/ipfs/⟨CID⟩/path, resolución del CID mediante DHT para encontrar providers (nodos que tienen el contenido), recuperación de contenido conectando a providers vía Bitswap/Graphsync, descarga de bloques del Merkle DAG y reconstrucción del archivo, y entrega HTTP estándar al navegador del usuario.

Los gateways públicos (ej. ipfs.io, dweb.link) proporcionan acceso conveniente pero introducen puntos de confianza y posible congestión. Los gateways privados/dedicados, ejecutados por organizaciones dentro de su red privada, ofrecen acceso controlado con código confiable. Los subdomain gateways (https://⟨CID⟩.ipfs.⟨gateway_url⟩/path) mejoran seguridad cross-origin (CORS) aislando cada CID en su propio subdominio. Los gateways simplifican el acceso (no requieren software especial), proporcionan puente entre HTTP e IPFS ($\leftrightarrow$), y el cacheo inteligente reduce latencia y uso de ancho de banda, aunque introducen limitaciones de confianza donde gateways públicos pueden censurar contenido o registrar actividad.

Integración con Blockchain y BlockDAG: Off-Chain Storage y On-Chain Metadata

El almacenamiento de datos grandes on-chain en blockchains tradicionales es costoso (gas fees altos), ineficiente (ledger inflado) y no escalable, dado que blockchains como Ethereum tienen límites de throughput (15-30 TPS) y tamaño de bloque restringido [19, 20]. La arquitectura híbrida combina almacenamiento off-chain en IPFS (para datos grandes como archivos, imágenes, videos, datasets) con registro on-chain de metadatos (solo el CID y metadatos críticos como propietario, timestamp, permisos) en blockchain o BlockDAG, manteniendo el ledger ligero y económico [19, 21].

El mecanismo de interacción estándar sigue el siguiente patrón: (1) Carga de datos: el usuario sube un archivo a IPFS (vía nodo local o pinning service), y IPFS devuelve el CID del contenido; (2) Registro on-chain: un smart contract almacena el CID en la blockchain junto con metadatos de propiedad/acceso; (3) Verificación: el CID on-chain actúa como "prueba de existenciaïnmutable, donde cualquier modificación del archivo off-chain generaría un CID diferente, detectándose la manipulación; (4) Recuperación: el usuario lee el CID desde la blockchain, luego descarga el contenido desde IPFS usando ese CID (vía gateway o nodo local).

La Figura 2.4 ilustra la arquitectura híbrida Blockchain-IPFS, mostrando el flujo completo desde la carga de datos hasta la recuperación verificable.

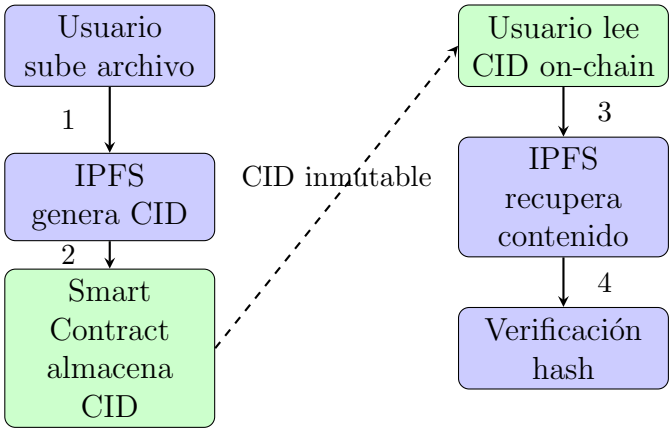

Figura 2.4: Arquitectura híbrida Blockchain-IPFS: flujo de carga (izquierda) y recuperación (derecha) con verificación criptográfica.

Esta integración ha sido documentada en múltiples casos de uso: NFTs (Non-Fungible Tokens) almacenan metadatos y arte digital en IPFS con CID referenciado en tokenURI de contratos ERC-721/ERC-1155, garantizando que el asset del NFT sea permanente y verificable; sistemas de Healthcare IoMT almacenan datos encriptados en clusters IPFS mientras smart contracts registran hashes para trazabilidad y privacidad [22]; verificación de documentos (diplomas, certificados) almacenados como archivos IPFS con blockchain manteniendo registro de autenticidad.

Los beneficios de la integración incluyen escalabilidad (reducción drástica del tamaño de transacciones on-chain, con ratios de compresión reportados de 0.0817) [3], reducción de costos (gas fees reducidos al almacenar solo hashes de 32-46 bytes vs. archivos completos), integridad (content-addressing de IPFS + inmutabilidad de blockchain proporcionan prueba criptográfica end-to-end), disponibilidad (IPFS distribuye réplicas vía BitSwap mientras blockchain asegura que el pointer CID sea permanente y resistente a censura), y deduplicación global (múltiples transacciones blockchain pueden referenciar el mismo CID IPFS sin duplicar almacenamiento).

Las limitaciones y consideraciones incluyen dependencia de pinning (si nadie ancla el contenido IPFS, puede volverse inaccesible aunque el CID esté on-chain, mitigado mediante incentivos económicos como Filecoin o SLAs con pinning services), latencia de recuperación (acceso a contenido IPFS puede ser más lento que almacenamiento centralizado debido a DHT lookup + block retrieval, mitigado mediante gateways con CDN), y asunciones de confianza (gateways

públicos introducen puntos de confianza, requiriendo nodos locales o gateways privados para operaciones más trustless).

Esta arquitectura híbrida es particularmente relevante para sistemas de monitoreo urbano que requieren almacenamiento frecuente de reportes de tráfico, configuraciones optimizadas y métricas históricas, donde la combinación de IPFS (alto throughput, deduplicación) y BlockDAG (baja latencia, costos reducidos) proporciona una solución escalable y económicamente viable para trazabilidad verificable en entornos de smart cities.

2.4. Lógica Difusa y Sistemas de Inferencia Mamdani

La lógica difusa y los sistemas de inferencia Mamdani proporcionan un marco formal para modelar razonamiento humano impreciso mediante variables lingüísticas, conjuntos difusos y reglas del tipo "SI-ENTONCES", y se han aplicado extensamente a la clasificación de estados de congestión vehicular en sistemas de transporte inteligente (ITS). Esta sección resume los fundamentos matemáticos de la lógica difusa, las funciones de membresía más utilizadas, la arquitectura del sistema Mamdani y ejemplos representativos de su uso en clasificación de tráfico.

Fundamentos de Lógica Difusa

La lógica difusa surge como una generalización de la lógica clásica binaria para tratar conceptos vagos como "bajo", "medio" o "alto", donde la pertenencia de un elemento a una clase no es dicotómica sino gradual. Zadeh define un conjunto difuso $\tilde{A}$ sobre un universo X mediante una función de membresía $\mu_{\tilde{A}} : X \to [0, 1]$, donde $\mu_{\tilde{A}}(x)$ indica el grado de pertenencia de x al conjunto [23]. Esta formalización permite extender nociones clásicas como unión, intersección y complemento a un contexto continuo, manteniendo propiedades de convexidad y posibilitando teoremas de separación para conjuntos difusos.

En lógica difusa, una variable lingüística es una variable cuyo valor son términos lingüísticos (por ejemplo, "velocidad" con valores {muy baja, baja, media, alta, muy alta}), y cada término se representa como un conjunto difuso sobre el dominio numérico correspondiente. Una proposición difusa, como "la velocidad es alta", se interpreta como un grado en $[0, 1]$ obtenido evaluando la función de membresía asociada al término "alta" en el valor numérico observado de la velocidad [24]. Esta capacidad de modelar gradualidad es fundamental en aplicaciones donde los conceptos no tienen límites precisos, como en la clasificación de estados de tráfico donde la transición entre "flujo libre" y "congestión" es continua y subjetiva.

Funciones de Membresía

Las funciones de membresía definen la forma de los conjuntos difusos e influyen directamente en la suavidad del razonamiento y en la sensibilidad del sistema. En aplicaciones de control e

ITS se utilizan habitualmente funciones de tipo triangular, trapezoidal y gaussiana, debido a su simplicidad y a la facilidad de parametrización.

La función de membresía triangular está definida por tres parámetros (a, b, c) que determinan el inicio, el pico y el final del soporte, y su expresión típica es:

$$\mu(x) = \begin{cases} 0, & x \leq a \\ \frac{x-a}{b-a}, & a < x \leq b \\ \frac{c-x}{c-b}, & b < x < c \\ 0, & x \geq c \end{cases} \tag{2.1}$$

Esta función proporciona una transición lineal suave con máximo igual a 1 en b y se usa extensamente cuando se requiere una implementación computacional ligera, como en sistemas embebidos para control de tráfico en tiempo real.

La función trapezoidal generaliza la triangular añadiendo una meseta central, y se define por cuatro parámetros (a, b, c, d) con:

$$\mu(x) = \begin{cases} 0, & x \leq a \\ \frac{x-a}{b-a}, & a < x \leq b \\ 1, & b < x \leq c \\ \frac{d-x}{d-c}, & c < x < d \\ 0, & x \geq d \end{cases} \tag{2.2}$$

Esta función permite modelar rangos donde el grado de pertenencia se considera plenamente verdadero en un intervalo y se degrada linealmente en los extremos. Cuando $b = c$, la trapezoidal se reduce a una triangular, proporcionando flexibilidad en el modelado de conceptos que tienen una zona central de certeza máxima.

La función de membresía gaussiana se caracteriza por parámetros c (centro) y σ (desviación estándar):

$$\mu(x) = \exp\left(-\frac{(x-c)^2}{2\sigma^2}\right) \tag{2.3}$$

Esta función produce una transición suave e infinitamente diferenciable, lo que resulta útil cuando se desean gradientes suaves o se combinan sistemas difusos con métodos de optimización continua. En sistemas de clasificación y control adaptativo, las gaussianas suelen proporcionar un ajuste más realista de distribuciones empíricas a costa de mayor complejidad de parametrización.

La Figura 2.5 ilustra cómo una variable numérica (velocidad promedio) se mapea a términos lingüísticos mediante funciones de membresía de diferentes tipos, mostrando la superposición de conjuntos difusos y cómo un valor crisp genera múltiples grados de pertenencia simultáneos.

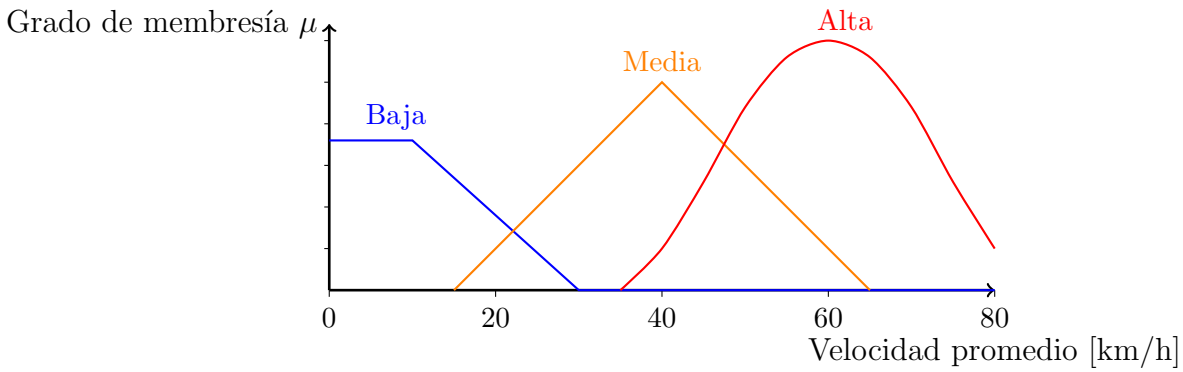

Figura 2.5: Variable lingüística *velocidad promedio* con tres conjuntos difusos (baja, media, alta) modelados mediante funciones de membresía trapezoidal, triangular y aproximadamente gaussiana.

Sistema de Inferencia Mamdani

El sistema de inferencia Mamdani fue introducido por Mamdani y Assilian como un enfoque para implementar controladores basados en reglas lingüísticas expresadas por expertos humanos [25]. Este tipo de sistema se caracteriza porque tanto los antecedentes como los consecuentes de las reglas son conjuntos difusos y porque la salida final se obtiene mediante un proceso de defuzzificación de un conjunto difuso agregado.

Un sistema Mamdani típico consta de cuatro etapas principales. La primera etapa es la fuzzificación, donde los valores de entrada crisp (numéricos) se transforman en grados de pertenencia a los conjuntos difusos definidos para cada variable lingüística de entrada. Para cada valor de entrada x_i y cada término lingüístico A_{ij}, se calcula $\mu_{A_{ij}}(x_i)$, generando un vector de pertenencias que alimenta el motor de inferencia.

La segunda etapa es la evaluación de reglas (inferencia local). Las reglas Mamdani suelen tener la forma:

$$\text{SI } x_1 \text{ es } A_{k1} \text{ Y } x_2 \text{ es } A_{k2} \text{ ENTONCES } y \text{ es } B_k \tag{2.4}$$

Los antecedentes se combinan mediante operadores t-norma (típicamente mínimo o producto) y t-conorma (máximo para OR) para obtener el grado de activación de cada regla. Este grado escala el conjunto difuso consecuente B_k, generando una salida difusa por regla mediante implicación min o producto.

La tercera etapa es la agregación de salidas. Las salidas difusas de todas las reglas se combinan en un único conjunto difuso de salida usando generalmente el operador máximo punto a punto sobre el universo de discurso de la variable de salida. El resultado es la salida agregada, que representa la contribución conjunta de todas las reglas del sistema para el vector de entrada dado.

La cuarta etapa es la defuzzificación. La salida agregada es un conjunto difuso que debe convertirse en un valor numérico (crisp) para su uso en control o decisión. El método de defuzzificación más empleado en sistemas Mamdani es el centroide o centro de gravedad (COG),

calculado como:

$$y^* = \frac{\int y\, \mu_{\text{out}}(y)\, dy}{\int \mu_{\text{out}}(y)\, dy} \tag{2.5}$$

donde μ_{out} es la función de membresía del conjunto de salida agregado. Otros métodos incluyen el máximo de la media (mean of maxima) o el máximo del máximo (argmax), pero el centroide suele ofrecer un compromiso adecuado entre estabilidad y sensibilidad.

El trabajo original de Mamdani demostró que un controlador difuso basado en reglas lingüísticas podía igualar o superar el desempeño de controladores clásicos para una planta de vapor, estableciendo la viabilidad práctica de este enfoque [25].

La Figura 2.6 muestra el flujo completo de un sistema Mamdani, desde las entradas numéricas hasta la salida crisp, ilustrando las cuatro etapas principales del proceso de inferencia.

La Figura 2.7 ilustra el funcionamiento de una regla individual, mostrando cómo los grados de pertenencia de las entradas se combinan para activar el consecuente difuso con un grado de activación determinado.

Aplicación en Clasificación de Congestión Vehicular

En sistemas de transporte inteligente, la lógica difusa se utiliza ampliamente para modelar la percepción humana de la congestión y para construir clasificadores que estimen el estado del tráfico a partir de variables macroscópicas como flujo, velocidad y densidad. La naturaleza gradual de conceptos como "flujo alto" o "velocidad baja" se adapta de forma natural al formalismo de variables lingüísticas y conjuntos difusos, evitando umbrales rígidos que pueden ser sensibles al ruido.

La Figura 2.8 muestra un ejemplo de integración de un sistema difuso Mamdani en un entorno ITS, ilustrando el flujo desde la captura de datos de sensores hasta la generación de recomendaciones para el control de tráfico.

Un esquema típico de clasificación difusa de congestión vehicular con sistema Mamdani incluye la selección de variables de entrada. Variables macroscópicas como flujo (vehículos/hora), velocidad media (km/h) y densidad (vehículos/km) son comúnmente utilizadas. Algunos trabajos consideran adicionalmente ocupación o tiempo de viaje para capturar mejor la variabilidad del tráfico.

El diseño de variables lingüísticas y funciones de membresía implica que cada variable se discretiza en términos lingüísticos como "bajo", "medio", "alto", "muy alto", representados con funciones de membresía triangulares o trapezoidales sobre rangos calibrados a partir de datos empíricos. La calibración suele hacerse con apoyo de expertos en tráfico o mediante técnicas de clustering como fuzzy c-means para determinar automáticamente los puntos de corte.

La definición de reglas Mamdani construye una base de reglas del tipo: "SI flujo es bajo Y velocidad es alta ENTONCES congestión es libre", o "SI flujo es alto Y velocidad es baja Y densidad es alta ENTONCES congestión es severa". Estas reglas reflejan conocimiento experto o patrones encontrados en los datos históricos, permitiendo capturar relaciones no lineales entre

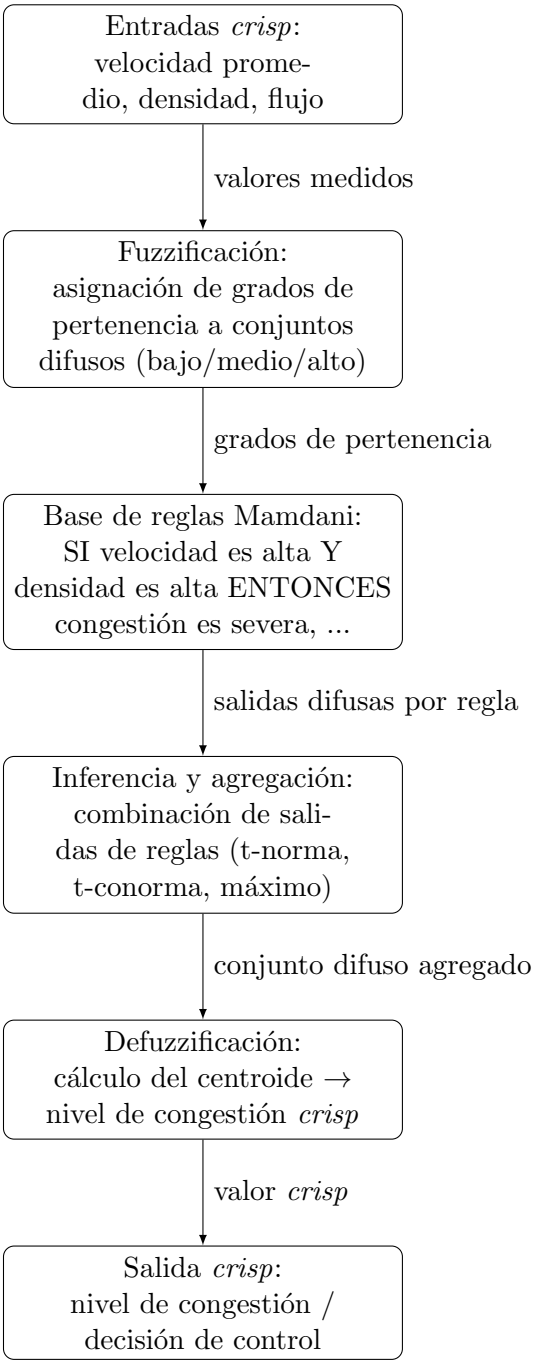

Figura 2.6: Flujo de un sistema de inferencia difusa tipo Mamdani: de entradas numéricas a una salida *crisp* mediante fuzzificación, reglas, agregación y defuzzificación.

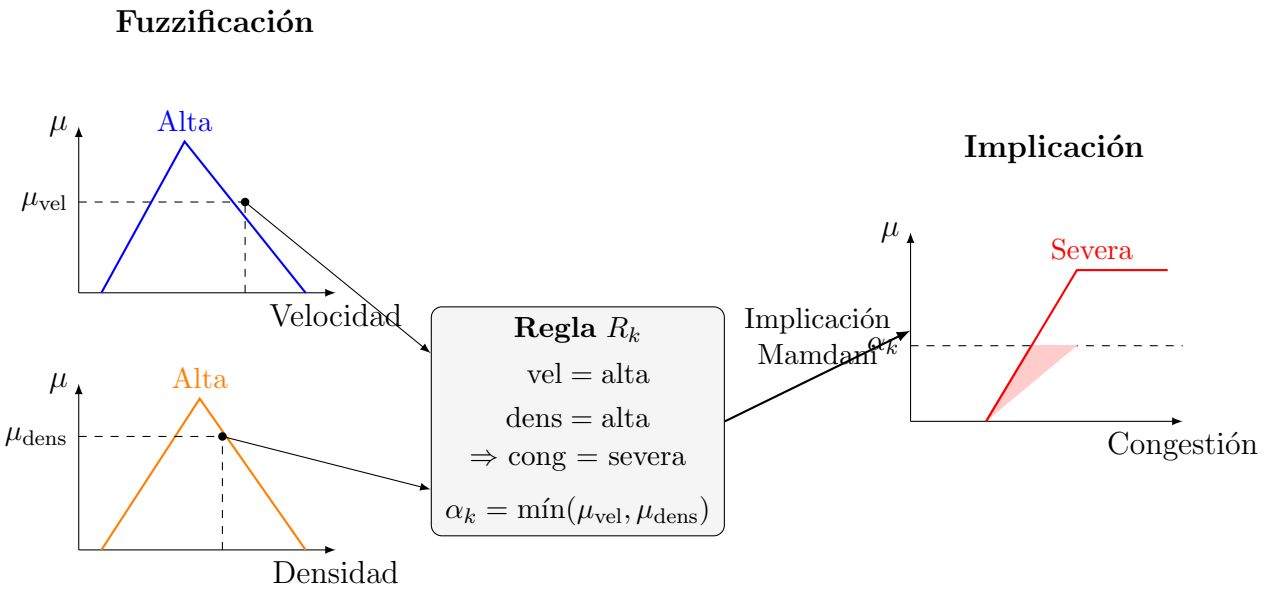

Figura 2.7: Inferencia difusa tipo Mamdani para una regla individual. Las entradas se fuzzifican, se combinan mediante un operador Y y activan el consecuente mediante recorte con grado α_k.

variables de tráfico que serían difíciles de modelar con técnicas deterministas.

El proceso de inferencia y clasificación de estado permite que el sistema Mamdani procese en tiempo (casi) real las mediciones procedentes de detectores o sensores y devuelva un nivel de congestión difuso que se defuzzifica a categorías discretas como {libre, estable, inestable, congestionado}. Estudios recientes muestran que estos clasificadores difusos son robustos ante datos incompletos o ruidosos y capturan de manera más fiel la percepción de los conductores respecto a soluciones crisp basadas en umbrales fijos [26, 27].

En el contexto de sistemas de gestión de tráfico urbano, los sistemas de inferencia difusa tipo Mamdani permiten transformar variables como velocidad promedio, densidad vehicular y flujo en categorías lingüísticas ("bajo", "medio", "alto") mediante funciones de membresía, y combinar reglas del tipo "si-entonces" para clasificar el grado de congestión en niveles como "leve", "moderada" o "severa", facilitando la toma de decisiones bajo incertidumbre inherente a la variabilidad del tráfico urbano.

La aplicación de lógica difusa Mamdani al control de semáforos y a la gestión dinámica de tráfico ha sido ampliamente documentada en la literatura. Diversos estudios reportan mejoras en tiempos de viaje y reducción en el número de paradas frente a esquemas de control fijo [26, 27, 28]. Por ejemplo, Erdinç propone un identificador difuso de estado de tráfico basado en Mamdani que utiliza flujo, velocidad y densidad como entradas y produce una clasificación continua del estado de congestión, demostrando que el modelo captura correctamente las transiciones entre flujo libre y condiciones hipercríticas [28]. Otros trabajos aplican lógica difusa Mamdani al control de semáforos y a la gestión dinámica de carriles, mostrando reducciones significativas en tiempos de viaje y número de paradas frente a esquemas de control fijo [26, 27]. Estas evidencias posicionan a la lógica difusa Mamdani como una herramienta valiosa en sistemas de optimización de tráfico urbano que requieren adaptabilidad y robustez ante condiciones

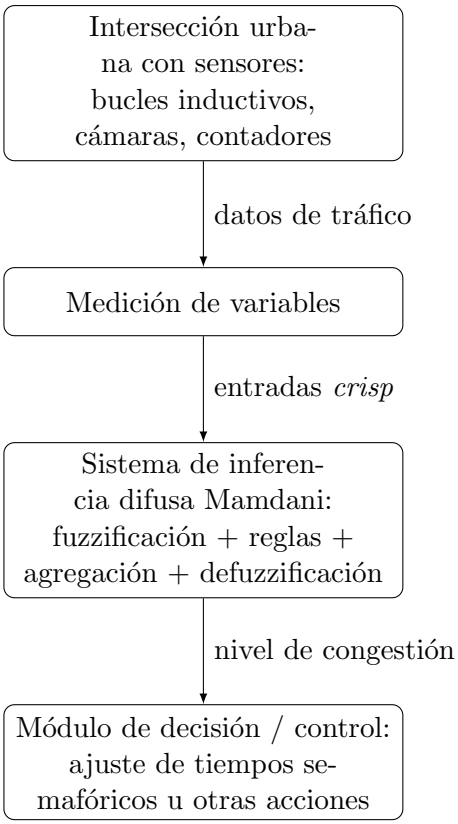

Figura 2.8: Ejemplo de integración de un sistema difuso Mamdani en un entorno ITS para clasificar congestión y alimentar un módulo de decisión o control semafórico.

variables.

2.5. Particle Swarm Optimization (PSO)

Particle Swarm Optimization (PSO) es un metaheurístico de optimización global inspirado en el comportamiento colectivo de bandadas de aves y bancos de peces, en el que un conjunto de partículas intercambia información sobre buenas soluciones para desplazarse en el espacio de búsqueda. En ingeniería de tráfico, PSO se ha aplicado con éxito al ajuste de tiempos semafóricos y otros parámetros de control, mostrando reducciones relevantes en demoras y tiempos de espera en simulaciones de intersecciones.

Algoritmo PSO: Inspiración y Dinámica Básica

PSO fue introducido por Kennedy y Eberhart en 1995 como un método para optimizar funciones no lineales mediante un conjunto de partículas que se mueven en un espacio de búsqueda multidimensional [29]. Cada partícula representa una solución candidata, caracterizada por una posición x_i y una velocidad v_i, que se actualizan en función de la experiencia propia y de la del grupo.

La inspiración biológica proviene de modelos de comportamiento social en bandadas, en los que cada individuo ajusta su trayectoria considerando tanto su mejor experiencia previa como

la de sus vecinos o la del enjambre completo. En PSO, esta idea se formaliza mediante dos componentes: el componente cognitivo, que representa la tendencia de la partícula a regresar a su mejor posición individual conocida (pbest), y el componente social, que representa la tendencia a acercarse a la mejor posición encontrada por el grupo (gbest) o por su vecindario en variantes locales [30].

La Figura 2.9 ilustra esquemáticamente el concepto del enjambre PSO en un espacio de búsqueda bidimensional, mostrando cómo las partículas se mueven influenciadas por su mejor experiencia personal y la mejor experiencia global del grupo.

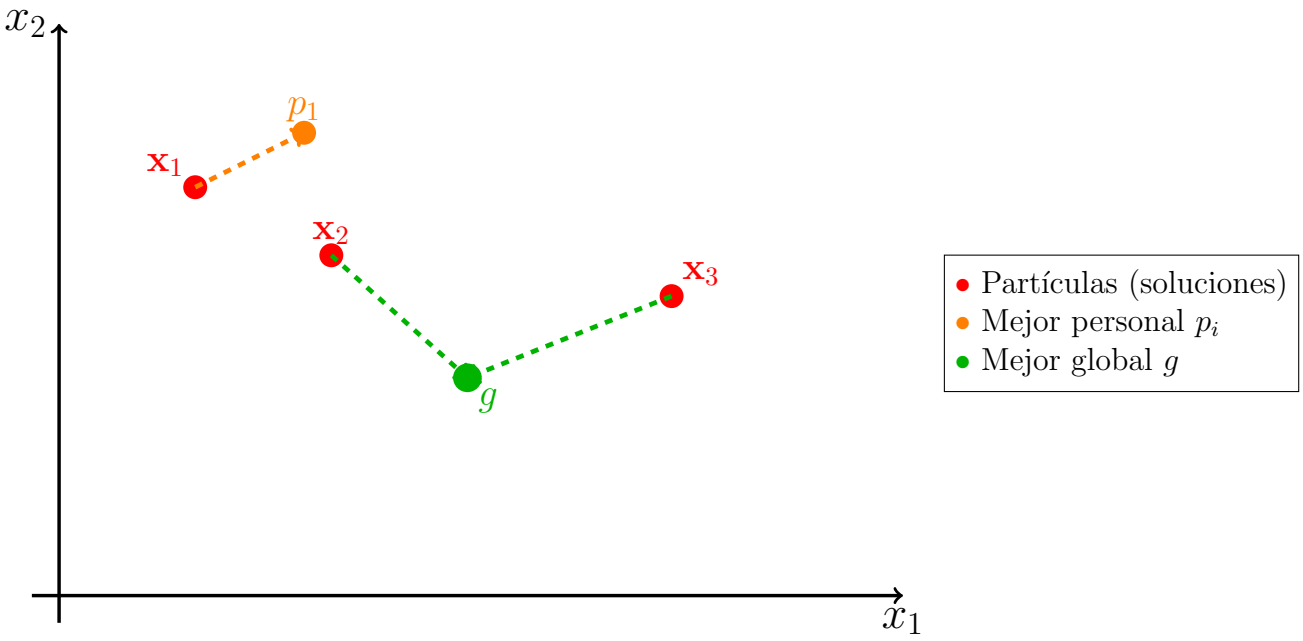

Figura 2.9: Esquema conceptual del enjambre PSO en un espacio de búsqueda bidimensional, mostrando partículas, mejor posición personal (p_i) y mejor posición global (g).

En su forma canónica (modelo global), las ecuaciones de actualización para la partícula i en la iteración t son:

$$v_i^{t+1} = w\, v_i^t + c_1 r_1 (p_i - x_i^t) + c_2 r_2 (g - x_i^t) \tag{2.6}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{2.7}$$

donde p_i es la mejor posición personal de la partícula, g es la mejor posición global del enjambre, $r_1, r_2 \sim U(0,1)$ son números aleatorios uniformes, w es el peso de inercia y c_1, c_2 son coeficientes de aceleración que ponderan la influencia del componente cognitivo y social, respectivamente [31].

El proceso iterativo incluye: (1) evaluar la función objetivo $f(x_i^t)$ para todas las partículas, (2) actualizar p_i si $f(x_i^t) < f(p_i)$, (3) actualizar g (global o local según la topología), y (4) actualizar velocidades y posiciones con las ecuaciones anteriores. Frecuentemente se impone un

límite de velocidad $|v_{id}| \leq v_{\text{máx},d}$ para controlar la amplitud de los saltos y evitar divergencia.

La Figura 2.10 muestra el diagrama de flujo del algoritmo PSO clásico, ilustrando el ciclo iterativo de evaluación, actualización y movimiento de partículas.

La Figura 2.11 ilustra la interpretación geométrica de la ecuación de actualización de velocidad, mostrando cómo los tres componentes (inercia, cognitivo y social) se combinan para determinar la nueva velocidad y posición de la partícula.

Parámetros Principales: Peso de Inercia, Coeficientes de Aceleración y Tamaño de Población

El comportamiento de PSO está fuertemente gobernado por cuatro parámetros clave: el peso de inercia w, los coeficientes de aceleración c_1, c_2 y el tamaño de la población (número de partículas).

El peso de inercia w controla cuánto de la velocidad anterior se preserva en cada actualización, modulando el equilibrio entre exploración y explotación. Valores grandes de w (p.ej. $w > 0{,}8$) permiten pasos más largos, favoreciendo la exploración global, pero pueden generar oscilaciones o divergencia si son excesivos. Valores pequeños de w (p.ej. $w < 0{,}4$) fomentan la explotación local y la convergencia fina, pero pueden provocar estancamiento en óptimos locales [32].

Shi y Eberhart propusieron un esquema de inercia decreciente, donde w disminuye linealmente desde un valor inicial alto (p.ej. 0.9) a uno bajo (p.ej. 0.4) a lo largo de las iteraciones, logrando una exploración inicial seguida de explotación refinada. Estudios recientes también introducen pesos de inercia adaptativos basados en medidas de diversidad del enjambre o aprendizaje automático, para ajustar w dinámicamente según el estado de búsqueda [31].

Los parámetros c_1 (cognitivo) y c_2 (social) ponderan la atracción hacia la mejor posición personal y la mejor posición global, respectivamente. Un c_1 alto refuerza el comportamiento individual, promoviendo una búsqueda más dispersa al permitir que cada partícula explore alrededor de su propia experiencia. Un c_2 alto incrementa la influencia del mejor miembro del grupo, favoreciendo una convergencia más rápida hacia la región más prometedora, pero con riesgo de convergencia prematura [31].

Análisis de estabilidad muestran que la suma $\phi = c_1 + c_2$ juega un papel crítico. La formulación con factor de constricción de Clerc y Kennedy utiliza:

$$\chi = \frac{2}{\phi - 2 + \sqrt{\phi^2 - 4\phi}}, \quad \text{con} \quad \phi > 4 \tag{2.8}$$

y redefine la ecuación de velocidad como:

$$v_i^{t+1} = \chi \left(v_i^t + c_1 r_1 (p_i - x_i^t) + c_2 r_2 (g - x_i^t) \right) \tag{2.9}$$

para garantizar convergencia estable del enjambre. En la práctica, se emplean con frecuencia valores $c_1 \approx c_2 \in [1{,}5, 2{,}5]$, manteniendo ϕ en un rango moderado [31].

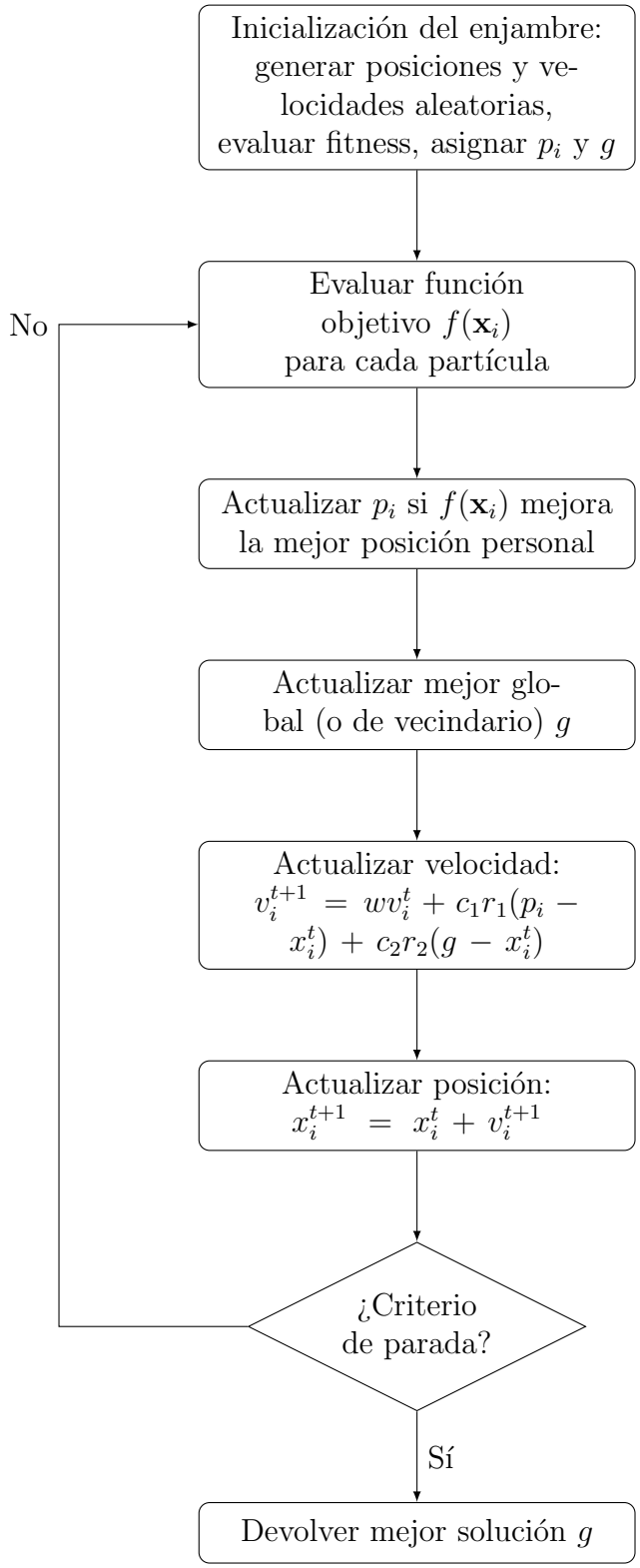

Figura 2.10: Diagrama de flujo del algoritmo PSO clásico, mostrando el ciclo iterativo de evaluación, actualización y movimiento de partículas.

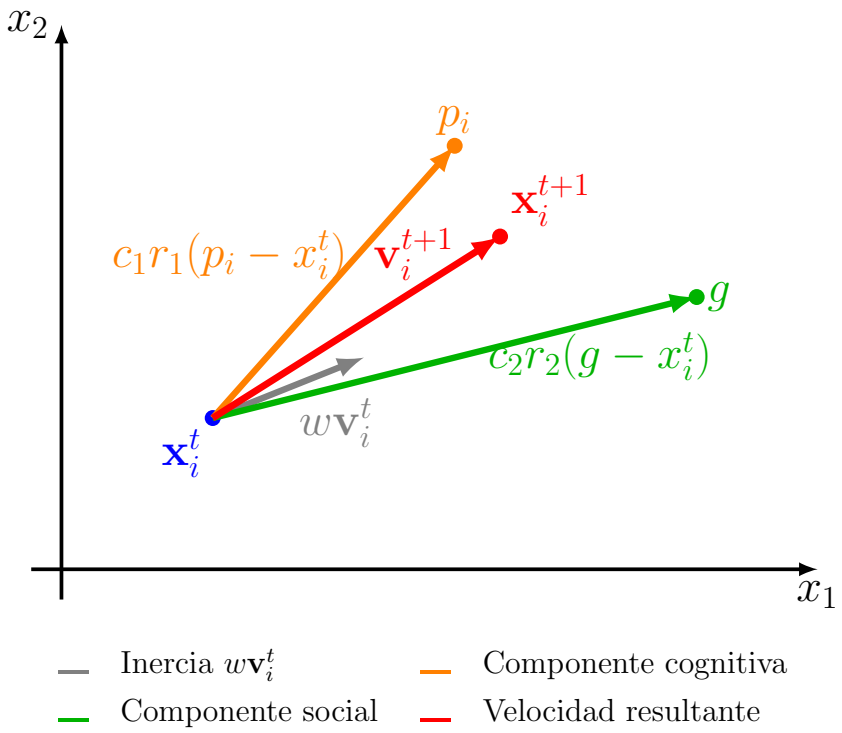

Figura 2.11: Interpretación geométrica de la actualización de velocidad en PSO.

El número de partículas determina el grado de muestreo del espacio de búsqueda y el costo computacional por iteración. Poblaciones pequeñas reducen el costo computacional, pero pueden cubrir de forma insuficiente el espacio y tienen mayor probabilidad de caer en óptimos locales. Poblaciones grandes mejoran la diversidad y la probabilidad de encontrar zonas promisorias, a costa de mayor tiempo de cómputo por iteración. Estudios empíricos sugieren que tamaños de enjambre en el rango de 20–60 partículas son adecuados para muchos problemas de dimensión moderada, mientras que problemas de alta dimensión o paisajes muy multimodales pueden requerir enjambres mayores o estrategias de partición del enjambre [33].

Convergencia y Optimización Global

Aunque PSO se diseñó originalmente como un algoritmo heurístico, se ha desarrollado una extensa literatura sobre su análisis de convergencia y propiedades de optimización global. Al interpretar las ecuaciones de actualización como un sistema dinámico discreto, es posible estudiar condiciones bajo las cuales las trayectorias de las partículas convergen [34].

Análisis lineales han mostrado que la estabilidad depende de la tripleta (w, c_1, c_2) y que ciertas combinaciones conducen a oscilaciones acotadas, convergencia a puntos fijos o divergencia. El uso de un factor de constricción χ y restricciones en la magnitud de la velocidad (clamping) son técnicas habituales para evitar explosiones de la velocidad y asegurar trayectorias acotadas [31].

Van den Bergh y Engelbrecht propusieron una versión modificada de PSO con garantías de convergencia a mínimos locales, identificando condiciones en las que el enjambre no se estanca en estados degenerados y proponiendo extensiones con convergencia global. Más recientemente,

enfoques de tipo mean-field han demostrado convergencia en probabilidad hacia el óptimo global bajo hipótesis de regularidad y crecimiento de la función objetivo [34, 35].

PSO es un método de optimización sin gradiente (zero-order), lo que lo hace adecuado para funciones no suaves, no derivables o ruidosas. La exploración global se logra mediante la combinación de inicialización aleatoria amplia del enjambre, componentes estocásticas r_1, r_2, y balance adecuado entre w, c_1 y c_2. Sin embargo, al igual que otros metaheurísticos, PSO no garantiza en general encontrar el óptimo global en un número finito de iteraciones para funciones arbitrarias, por lo que se habla de convergencia en probabilidad o convergencia bajo supuestos específicos [33].

Para mejorar la capacidad de escape de óptimos locales, se han propuesto variantes como PSO con topologías de vecindario local (lbest) en lugar de gbest, PSO híbrido con mutación tipo algoritmos genéticos o reinicialización parcial de partículas, y PSO multi-objetivo y multi-enjambre (multi-swarm) para cubrir diversos modos del paisaje [33, 31].

Aplicaciones en Optimización de Tráfico y Tiempos Semafóricos

En el ámbito de sistemas de transporte inteligente, PSO se ha utilizado para optimizar planes de señalización semafórica, coordinación de fases y otros parámetros de control de tráfico en redes urbanas. La idea central es codificar los tiempos de verde/ámbar/rojo (y eventualmente offsets entre intersecciones) como variables de decisión en las partículas, y minimizar funciones objetivo relacionadas con el desempeño del tráfico.

Para el caso de optimización de tiempos de semáforos en una intersección o red pequeña, una partícula suele codificar un vector del tipo $x_i = (g_1, g_2, \ldots, g_M, \text{offset}_1, \ldots)$, donde g_k es el tiempo de verde de la fase k, y opcionalmente se incluyen offsets o tiempos intermedios. Se imponen restricciones como límites mínimos y máximos por fase ($g_k^{\text{mín}} \leq g_k \leq g_k^{\text{máx}}$) y suma de tiempos por ciclo $\sum_k g_k = C$ o dentro de un rango $[C_{\text{mín}}, C_{\text{máx}}]$. Estas restricciones se pueden manejar mediante proyección de x_i al espacio factible después de cada actualización o penalización en la función objetivo si la solución viola restricciones.

La función objetivo puede ser minimizar el tiempo medio de retraso por vehículo, minimizar la suma ponderada de colas, retrasos y emisiones, o maximizar un índice de nivel de servicio (LOS). Para cada partícula, se configura el plan semafórico en el simulador (p.ej. SUMO, VISSIM), se ejecuta la simulación para un periodo representativo, y se mide el valor de la función objetivo. Dado que cada evaluación es costosa, se suele utilizar poblaciones moderadas (20–40 partículas) y número limitado de iteraciones (p.ej. 30–50).

Trabajos como el de Li et al. proponen enfoques de PSO para la combinación de fases y tiempos de verde en intersecciones, minimizando el número de paradas y el retraso total [36]. En un estudio sobre una intersección, un modelo PSO para combinación de fases logró una reducción aproximada del 19 % en el número de paradas respecto a un plan fijo base, junto con reducciones simultáneas en retraso total y emisiones de CO derivadas del menor número de arranques y frenadas.

Gonçalo y colaboradores plantean un esquema de simulación–optimización donde PSO se integra con el simulador SUMO para ajustar los tiempos de luz verde en problemas de control de tráfico [37]. Cada partícula representa una configuración de tiempos de fase para los semáforos. Para cada configuración, SUMO simula el tráfico y devuelve métricas como tiempo medio de espera y número de vehículos en cola. PSO actualiza las partículas para minimizar el tiempo medio de espera. Los resultados reportan reducciones cercanas al 26 % en el tiempo medio de espera respecto a configuraciones iniciales, demostrando que PSO puede encontrar esquemas de señalización más eficientes que configuraciones estáticas o heurísticas simples.

En la literatura de ITS se han explorado variantes de PSO adaptadas al contexto de tráfico, por ejemplo: PSO multiobjetivo para equilibrar retraso, emisiones y equidad entre movimientos; PSO combinado con lógica difusa, donde PSO ajusta parámetros de control difuso (p. ej. funciones de membresía de un controlador de semáforos) para mejorar la calidad de servicio del tráfico; y PSO distribuido para redes grandes, donde distintos subenjambres optimizan regiones de la red de forma coordinada [37, 36].

En conjunto, estos resultados posicionan a PSO como una herramienta flexible y robusta para la optimización de control de tráfico, especialmente en escenarios donde la función objetivo es evaluada mediante simulación y carece de gradiente analítico, como ocurre en sistemas de optimización de tráfico urbano que requieren integración con simuladores microscópicos para evaluar el desempeño de configuraciones de semáforos.

CAPÍTULO 3

Capitulo

CAPÍTULO 4

capitulo

CAPÍTULO 5

CONCLUSIÓN

REFERENCIAS

[1] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. Consultado: 2025-01-XX. 2008. URL: https://bitcoin.org/bitcoin.pdf.

[2] Juan Garay, Aggelos Kiayias y Nikos Leonardos. «The Bitcoin Backbone Protocol: Analysis and Applications». En: *Proceedings of the 19th International Conference on Principles of Distributed Systems (OPODIS 2015)*. Consultado: 2025-01-XX. 2015, págs. 1-15. URL: https://opodis2015.irisa.fr/files/2016/01/Garay_OPODIS_2015.pdf.

[3] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen y Huaimin Wang. «Blockchain challenges and opportunities: a survey». En: *International Journal of Web and Grid Services* 14.4 (2018), págs. 352-375. DOI: 10.1504/IJWGS.2018.095647.

[4] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miller y Steven Goldfeder. *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction*. Princeton University Press, 2016. ISBN: 978-0-691-17169-2.

[5] Vitalik Buterin. «Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform». En: (2014). Whitepaper, Consultado: 2025-01-XX. URL: https://ethereum.org/en/whitepaper/.

[6] Matteo Belotti, Nikola Bozic, Guy Pujolle y Stefano Secci. «A Vademecum on Blockchain Technologies: When, Which, and How». En: *IEEE Communications Surveys & Tutorials* 21.4 (2019). Revisión sistemática sobre escalabilidad blockchain, págs. 3796-3838. DOI: 10.1109/COMST.2019.2928178.

[7] Marco Conoscenti, Antonio Vetro y Juan Carlos De Martin. «Blockchain for the Internet of Things: A Survey». En: *2016 IEEE 19th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*. IEEE, 2016, págs. 1-6. DOI: 10.1109/SNPD.2016.7515925.

[8] Florian Eckermann, Hendrik Beckmann, Benjamin Hagedorn y Marcus Kaulartz. «Blockchain Technology in the Energy Sector: A Systematic Review of Challenges and Opportunities». En: *Renewable and Sustainable Energy Reviews* 100 (2019). Análisis de costos y limitaciones de blockchain, págs. 143-174. DOI: 10.1016/j.rser.2018.10.014.

[9] Qiang Wang, Jiajing Yu, Shuo Chen y Yong Xiang. «SoK: DAG-based Blockchain Systems». En: *ACM Computing Surveys* 55.12 (2023), Article 256. DOI: 10.1145/3576899.

[10] Yonatan Sompolinsky y Aviv Zohar. «PHANTOM and GHOSTDAG: A Scalable Generalization of Nakamoto Consensus». En: *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies (AFT '21)*. New York, NY, USA: ACM, 2021, págs. 57-70. DOI: 10.1145/3479722.3480990.

[11] Vitalik Buterin. *On Public and Private Blockchains*. Origen del concepto "blockchain trilemma"sobre seguridad, descentralización y escalabilidad. 2015. URL: https://blog.ethereum.org/2015/08/07/on-public-and-private-blockchains.

[12] Qiang Wang, Shuo Li, Xuanzhe Luo et al. «SoK: Diving into DAG-based Blockchain Systems». En: *IEEE Communications Surveys & Tutorials* (2020). Systematization of Knowledge sobre sistemas blockchain basados en DAG y mitigación del blockchain trilemma. arXiv: 2012.06128 [cs.CR]. URL: https://arxiv.org/abs/2012.06128.

[13] Bitcoin Core Developers. *Bitcoin Protocol Documentation*. Documentación técnica sobre block time de 10 minutos en Bitcoin. 2024. URL: https://en.bitcoin.it/wiki/Block.

[14] Ethereum Foundation. *Blocks*. Documentación oficial de Ethereum sobre block time de 12 segundos en PoS. 2024. URL: https://ethereum.org/developers/docs/blocks/.

[15] Kaspa Team. *Kaspa: A GhostDAG Implementation*. Implementación práctica de GHOSTDAG con block time sub-segundo y alto throughput. 2021. URL: https://kaspa.org/publications/.

[16] X. Zhang et al. «SoK: DAG-based Consensus Protocols». En: *arXiv preprint* (2025). Systematization of Knowledge sobre protocolos de consenso basados en DAG. arXiv: 2411.10026 [cs.CR]. URL: https://arxiv.org/abs/2411.10026.

[17] Juan Benet. *IPFS - Content Addressed, Versioned, P2P File System*. Whitepaper fundacional de IPFS. 2014. arXiv: 1407.3561 [cs.DC]. URL: https://arxiv.org/abs/1407.3561.

[18] IPFS Project. *IPFS Architecture*. Especificación oficial de arquitectura IPFS, define formalmente nodeID := multihash(publicKey) y esquema de identidad basado en PKI. 2024. URL: https://github.com/ipfs/specs/blob/main/ARCHITECTURE.md.

[19] Fei Yang, Wei Zhou, Qiang Wu, Ruyin Long, Neal N. Xiong y Mengchu Zhou. «Interaction mechanism between blockchain and IPFS». En: *ELS Publishing* 1.2 (2023). DOI: 10.37293/ELS.2023.002.

[20] T. S. Vasughi, P. Muthulakshmi y D. Gritto. «Off-chain Data Storage Blockchain using Interplanetary File System». En: *International Journal of Scientific Development and Research (IJSDR)* 8.3 (2023), págs. 40-44.

[21] Md. Rashedul Haque et al. «An integrated blockchain and IPFS-based solution for secure and traceable code repository management». En: *PLOS ONE* 19.9 (2024), e0331131. DOI: 10.1371/journal.pone.0331131.

[22] Rakesh Kumar et al. «Blockchain and IPFS-based Healthcare Data Management System». En: *IEEE Transactions on Network and Service Management* (2020). Aplicación de IPFS en Healthcare IoMT.

[23] Lotfi A. Zadeh. «Fuzzy Sets». En: *Information and Control* 8.3 (1965), págs. 338-353. DOI: 10.1016/S0019-9958(65)90241-X.

[24] Lotfi A. Zadeh. «Similarity Relations and Fuzzy Orderings». En: *Information Sciences* 3.2 (1971), págs. 177-200. DOI: 10.1016/S0020-0255(71)80005-1.

[25] Ebrahim H. Mamdani y Sedrak Assilian. «An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller». En: *International Journal of Man-Machine Studies* 7.1 (1975), págs. 1-13. DOI: 10.1016/S0020-7373(75)80002-2.

[26] Rizka Zuliani Musriroh. «Application of Mamdani Method on Fuzzy Logic to Decision of Traffic Light System». En: *Proceedings of International Conference on Informatics and Computational Sciences*. SciTePress, 2020, pág. 85200. DOI: 10.5220/0008520000000000.

[27] M. Esmaeili et al. «An Optimization Similarity Fuzzy Inference Method for Traffic Signal Control». En: *Journal of Intelligent Transportation Systems* (2025). En prensa.

[28] Gökhan Erdinç. «Application of a Mamdani-Based Fuzzy Traffic State Identifier». Tesis sobre identificación de estado de tráfico usando lógica difusa Mamdani. Thesis. Università "La Sapienza"di Roma, 2023. URL: https://iris.uniroma1.it/retrieve/d51daa72-738b-40a8-b0f8-2f25473ebf1b/Erdin%C3%A7_Application-Mamdani-based_2023.pdf.

[29] James Kennedy y Russell Eberhart. «Particle Swarm Optimization». En: *Proceedings of the IEEE International Conference on Neural Networks*. Vol. 4. IEEE, 1995, págs. 1942-1948. DOI: 10.1109/ICNN.1995.488968.

[30] Russell C. Eberhart y James Kennedy. «A New Optimizer Using Particle Swarm Theory». En: *Proceedings of the Sixth International Symposium on Micro Machine and Human Science (MHS '95)*. IEEE, 1995, págs. 39-43. DOI: 10.1109/MHS.1995.494215.

[31] Maurice Clerc y James Kennedy. «The Particle Swarm – Explosion, Stability, and Convergence in a Multidimensional Complex Space». En: *IEEE Transactions on Evolutionary Computation* 6.1 (2002), págs. 58-73. DOI: 10.1109/4235.985692.

[32] Yuhui Shi y Russell Eberhart. «Parameter Selection in Particle Swarm Optimization». En: (1998). Proposed inertia weight adaptation, págs. 591-600.

[33] Riccardo Poli, James Kennedy y Tim Blackwell. «Particle Swarm Optimization: An Overview». En: *Swarm Intelligence* 1.1 (2007), págs. 33-57. DOI: 10.1007/s11721-007-0002-0.

[34] Frans Van den Bergh y Andries P. Engelbrecht. «A Convergence Proof for the Particle Swarm Optimiser». En: *Fundamenta Informaticae* 105.4 (2010), págs. 341-374. DOI: 10.3233/FI-2010-370.

[35] H. Chen et al. «Convergence Analysis of Particle Swarm Optimization with Stochastic Disturbances». En: *Frontiers in Applied Mathematics and Statistics* 10 (2024), pág. 1304268. DOI: 10.3389/fams.2024.1304268.

[36] X. Li et al. «A PSO Based Signal Timing Optimization Approach of Phase Combination». En: *Proceedings of the International Conference on Computer Technology and Engineering*. Aplicación de PSO a optimización de tiempos semafóricos. Atlantis Press, 2016, págs. 116-120.

[37] R. Gonçalo et al. «On Tuning the Particle Swarm Optimization for Solving the Traffic Light Time Problem». En: *ICCSA 2022 – Lecture Notes in Computer Science*. PSO integrado con SUMO para optimización de semáforos. Springer, 2022.

APÉNDICE

A.1. A