UNIVERSIDAD NACIONAL DE SAN MARTIN

RCSI
https://revistas.unsm.edu.pe/index.php/rcsi
e-ISSN: 2709-992X

Original article
Articulo original
Jul-Dec, 2026

# Cloud-based architecture of digital twins applied to the optimal design of large-scale supply chains

## Arquitectura cloud-based de gemelos digitales para el diseño óptimo de cadenas de suministro de gran escala.

**Marmolejo-Saucedo, José Antonio[1]***

**Rodríguez-Aguilar, Román[2]**

**Abraham Mendoza[1]**

[1]Facultad de Ingeniería, Universidad Panamericana, Álvaro del Portillo 49, Zapopan, Jalisco, 45010, Mexico
[2]Facultad de Ciencias Económicas y Empresariales, Universidad Panamericana, Augusto Rodin 498, Ciudad de Mexico, 03920, Mexico

**Recibido:** 18 May. 2026 | **Aceptado:** 11 Jul. 2026 | **Publicado:** 20 Jul. 2026

**Autor de correspondencia*:** jmarmolejo@up.edu.mx

**Cómo citar este artículo:** Marmolejo-Saucedo, J. A., Rodriguez-Aguilar, R. & Mendoza, A. (2026). Cloud-based architecture of digital twins applied to the optimal design of large-scale supply chains. *Revista Científica de Sistemas e Informática, 6*(2), e1682. https://doi.org/10.51252/rcsi.v6i2.1682

## ABSTRACT

The design of large-scale supply networks is an NP-hard MILP problem that requires dynamic representation and efficient optimization in industrial contexts. This study proposed, implemented, and validated a four-layer cloud-based architecture that integrates Microsoft Azure Digital Twins (ADT), a multi-echelon MILP model, a hybrid PSO, and a Python-based integration engine with bidirectional synchronization. The main algorithmic contribution consisted of using the linear relaxation of the MILP as the PSO fitness function, providing a formal lower bound for the original problem. A total of 74 experimental runs were conducted using a capacity repair operator, achieving a zero-infeasibility rate. The evaluated instances, across S, M, and XL scales and five complexity levels, showed an inverse relationship between the integrality gap and the quality of the PSO solution. Likewise, instances with low combinatorial complexity achieved over 90% consistency in opening decisions, preserving the decision state of the digital twin. The proposed framework integrates cloud-based architecture, MILP optimization, and hybrid PSO for supply chains with experimental validation.

**Keywords:** Azure Digital Twins; supply chain; network design; digital twin; PSO, MILP; Python

## RESUMEN

El diseño de redes de suministro de gran escala es un problema MILP NP-duro que requiere representación dinámica y optimización eficiente en contextos industriales. Este estudio propuso, implementó y validó una arquitectura cloud-based de cuatro capas que integra Microsoft Azure Digital Twins (ADT), un modelo MILP multi-escalón, un PSO híbrido y un motor de integración en Python con sincronización bidireccional. La principal contribución algorítmica consistió en emplear la relajación lineal del MILP como función de aptitud del PSO, proporcionando una cota inferior formal del problema original. Se ejecutaron 74 corridas experimentales con un operador de reparación por capacidad, alcanzando una tasa de infactibilidad nula. Las instancias evaluadas, en escalas S, M y XL y con cinco niveles de complejidad, evidenciaron una relación inversa entre la brecha de integralidad y la calidad de la solución PSO. Asimismo, las instancias de baja complejidad combinatoria lograron una consistencia superior al 90% en las decisiones de apertura, preservando el estado decisional del gemelo digital. El marco propuesto integra arquitectura cloud-based, optimización MILP y PSO híbrido para cadenas de suministro con validación experimental.

**Palabras clave:** Azure Digital Twins; cadena de suministro; diseño de red; gemelo digital; MILP; PSO; Python

## 1. INTRODUCTION

Large-scale supply chains constitute logistics systems with a high degree of combinatorial complexity. These systems are characterized by the interaction of multiple echelons or facility levels, including suppliers, production plants, distribution centers, and end customers; material and information flows; heterogeneous capacity constraints; and demand subject to variability and uncertainty (Simchi-Levi et al., 2008). Network design involves strategic decisions concerning facility location, opening and closure, flow allocation across echelons, and the determination of service levels (Pérez-Baquero et al., 2026). These mathematical models are NP-hard, and the search for optimal solutions exceeds the capabilities of conventional exact methods (Geoffrion & Graves, 1974; Melo et al., 2009).

Today, disruptive phenomena such as global pandemics, geopolitical volatility, demand fluctuations, and supplier-chain disruptions have highlighted the limitations of static network design models. Several authors addressing digital twins for supply chains have emphasized the potential of cloud-based platforms to enable dynamic decision-making. Marmolejo-Saucedo (2020) proposes the development of digital twins through a case study employing specialized supply chain design software, while Marmolejo-Saucedo (2022) extends this approach to large-scale optimization problems. The development of digital twins based on this type of architecture enables the representation and management of facilities, as well as the automation of critical processes, thereby establishing an interconnected digital ecosystem capable of responding rapidly to environmental demands (Dominguez Martinez et al., 2026; Wasi et al., 2025).

A digital twin is a virtual and dynamic representation of a physical or logical system that is synchronized with its real-world counterpart through bidirectional data flows. This makes it possible to monitor the system's current state, simulate future scenarios, and provide feedback to operational contingency plans (Grieves, 2014; Tao et al., 2019). In the context of supply chains, digital twins offer the possibility of modeling network topology, operational parameters, and interdependencies among nodes while systematically exchanging and updating data (Ivanov & Dolgui, 2021). Microsoft Azure Digital Twins (ADT) is a platform as a service (PaaS) based on a language known as the Digital Twins Definition Language (DTDL). It currently represents one of the most mature cloud-based solutions available on the market and enables the integration of multiple information systems and technologies (Microsoft, 2025c).

The multi-echelon supply chain design problem is commonly formulated as a mixed-integer linear programming (MILP) model (Farahani et al., 2014; Melo et al., 2009). Problems of this type involving industrial-scale instances impose considerable computational complexity on exact solvers, as they may require substantial computation times (Avella et al., 2023). In this context, hybrid particle swarm optimization (PSO) has been reported to achieve significant reductions in computation time (Kennedy & Eberhart, 1995; Noel & Jannett, 2004). However, studies that employ PSO to solve supply chain MILP problems do not consider integration with cloud-based digital twin platforms.

The literature review suggests the existence of a technical gap, as no unified technological architecture formally integrates: (i) the ontological and dynamic representation of a multi-echelon supply chain; (ii) a MILP model directly coupled with this representation; and (iii) a hybrid PSO algorithm incorporating feasibility-repair mechanisms and performance evaluation based on the linear programming (LP) relaxation of the problem. Therefore, the main contributions of this study are: (1) a four-layer architecture illustrated in Figure 1; (2) a hybrid PSO algorithm that provides a bound with respect to the optimal solution, whose pseudocode is presented in Figure 10; and (3) experimental validation through multiple runs using instances of varying levels of complexity.

## 2. THEORETICAL FOUNDATIONS

### 2.1. Digital Twins in Supply Chain Management.

The digital twin concept was formalized in the field of manufacturing (Grieves, 2014; Grieves & Vickers, 2017), and its adoption in supply chain management has accelerated with the convergence of the Internet of Things (IoT), cloud computing, and real-time analytics (Lim et al., 2020; Tao et al., 2019). Studies in this field include the design of digital twins for supply chains (Marmolejo-Saucedo, 2020) and their extension to large-scale optimization problems, such as the packing problem (Marmolejo-Saucedo, 2022). These contributions established the methodological foundations for the integration of a cloud-based platform with a MILP problem proposed in the present study.

Ivanov and Dolgui (2021) distinguish three archetypes of digital twins in supply chain management: visibility twins, analytical twins, and optimization twins. The taxonomy proposed by Kritzinger et al. (2018) distinguishes among the "digital model," involving manual synchronization; the "digital shadow," involving unidirectional digital data flow; and the "digital twin," involving bidirectional synchronization. The proposed architecture implements the third category, operationalized through the *openStatus* property of the *Plant* and *DC* nodes in the ADT graph (see Figure 7). Ivanov and Dolgui (2019) and Longo et al. (2023) document the benefits of digital twins for supply chain resilience and the evolution toward integrated ecosystems. Cloud scalability is critical for managing the large volumes of data generated by large-scale supply chains (Somma et al., 2025; Wasi et al., 2025).

### 2.2. Microsoft Azure Digital Twins and DTDL modeling

Azure Digital Twins is a platform as a service (PaaS) that enables the creation of digital twin instance graphs and their querying through an SQL-like language (Microsoft, 2025c). Modeling is performed using DTDL v3 (Microsoft, 2023), a JSON-LD-based schema that defines properties, relationships, and telemetry. The relevant metamodel classes are *Interface*, *Property*, *Relationship*, and *Telemetry*. Integration with Azure IoT Hub, Event Grid, and Stream Analytics (Milenkovic, 2022; Somma et al., 2025) enables automatic synchronization in response to state changes. The proposed DTDL ontology for the supply chain, illustrated in Figure 3, has not been reported in the existing literature (Microsoft, 2025b), thus constituting an original contribution of this study.

### 2.3. Supply Network Optimization: MILP Models and Exact Approaches.

Geoffrion and Graves (1974) introduced the multi-echelon network design problem, which has subsequently been extensively reviewed (Farahani et al., 2014; Melo et al., 2009). Commercial solvers such as Gurobi, CPLEX, and CBC can solve medium-sized instances using branch-and-cut algorithms; however, their performance decreases exponentially as the number of binary variables increases (Avella et al., 2023). In this study, the LP relaxation provides a high-quality lower bound within polynomial computation time, which is used as the PSO fitness function.

### 2.4. Hybrid Metaheuristics for Combinatorial Optimization in Supply Chains.

The PSO algorithm was initially introduced by Kennedy and Eberhart (1995). Subsequently, Shi and Eberhart (1998) incorporated the inertia coefficient, thereby improving the algorithm's performance. The binary PSO variant, or BPSO (Kennedy & Eberhart, 1997), converts particle positions into binary values using a sigmoid function and is a standard technique for facility location and other discrete optimization problems. Noel and Jannett (2004) reported PSO–LP hybridizations that achieved significant performance improvements. In the present study, hybridization is implemented by evaluating fitness through the LP relaxation of the flow subproblem while facility-opening decisions remain fixed. This reduces the effective search space to only |I| + |J| variables. Guo and Mantravadi (2025) and Tirkolaee et al. (2020) document recent applications of PSO in supply chain optimization; however, these studies do not consider integration with cloud-based digital twin platforms.

## 3. MATERIALS AND METHODS.

### 3.1. Mathematical Formulation of the MILP Mode

#### 3.1.1. Sets, Parameters, and Variables

The mathematical model considers a three-echelon network comprising plants (I), CEDIS (J), and customers (K). Each plant (i) has a fixed opening cost $f_i^P$, a capacity $p_i$, and a variable production cost $v_i$. Each CEDIS (j) has a fixed opening cost $f_j^D$ and a capacity $s_j$, while each customer (k) demands $d_k$ units.

Transportation costs are denoted by $c_{ij}^{PD}$ between plants and CEDIS, and by $c_{jk}^{Dc}$ between CEDIS and customers. The model includes the following decision variables: $y_i \in \{0,1\}$, indicating whether plant (i) is opened; $z_j \in \{0,1\}$, indicating whether CEDIS is opened; $x_{ij} \geq 0$, representing flows from plants to CEDIS; and $q_{jk} \geq 0$, representing flows from CEDIS to customers. The ADT graph shown in Figure 6 stores these parameters as properties of the digital twin instances.

#### 3.1.2. Objective function

The objective function minimizes the total cost:

$$\min Z = \sum_{i\in I} f_i^P \cdot y_i + \sum_{j\in J} f_j^D \cdot z_j + \sum_{i\in I}\sum_{j\in J} \left(c_{ij}^{PD} + v_i\right) \cdot x_{ij} + \sum_{j\in J}\sum_{k\in K} c_{jk}^{DC} \cdot q_{jk}$$

#### 3.1.3. Constraints

The model incorporates the constraints typically employed in supply chain network design problems (Melo et al., 2009):

**(R1)** $\sum_{j\in J} q_{jk} = d_k \quad \forall k \in K$

**(R2)** $\sum_{i\in I} x_{ij} = \sum_{k\in K} q_{jk} \quad \forall j \in J$

**(R3)** $\sum_{j\in J} x_{ij} \leq p_i \cdot y_i \quad \forall i \in I$

**(R4)** $\sum_{k\in K} q_{jk} \leq s_j \cdot z_j \quad \forall j \in J$

**(R5)** $y_i \in \{0,1\}\ \forall i \in I,\ z_j \in \{0,1\}\ \forall j \in J,\ x_{ij} \geq 0,\ q_{jk} \geq 0.$

#### 3.1.4. Linear Relaxation and Its Role in PSO.

The LP relaxation replaces the binary conditions $y_i \in \{0,1\}$ and $z_j \in \{0,1\}$ with the continuous intervals [0,1]. The resulting value, $Z^{LP}$, provides a lower bound on $Z^{MIP}$, and this property justifies its use as the PSO fitness function. If a particle encodes the facility-opening vector $(\hat{y}, \hat{z})$, its actual cost can never be lower than $Z^{LP}(\hat{y}, \hat{z})$. In other words, the search guidance cannot be overly optimistic, a property that no heuristic fitness function can guarantee. The implementation solves these subproblems using Pyomo and the CBC/GLPK solvers (Hart et al., 2011).

$$Z^{LP}(\hat{y}, \hat{z}) \leq Z^{MIP}$$

### 3.2. Proposed Architecture: ADT–MILP–PSO Integration

#### 3.2.1. Overview of the Four-Layer Architecture

Figure 1 presents the complete architecture. It consists of four layers, each with clearly defined scope and boundaries: Azure Digital Twins (Layer 1), the Python execution engine (Layer 2), the MILP module (Layer 3), and the hybrid PSO algorithm (Layer 4). The layered design allows each component to remain independent and replaceable, for example, by changing the optimization solver or testing alternative PSO variants. The cloud infrastructure provides the scalability required by large-scale supply chains in terms of data volume and distributed operations (Marmolejo-Saucedo, 2020; Wasi et al., 2025).

Figure 2 represents the starting point of the implemented architecture. For this purpose, the Azure Digital Twins instance must have an **Active** status in the Azure Portal. Before executing any Python code, the Azure

cloud services must be accessible. The hostname displayed on the screen (`adt-scm-suministro.api.eus.digitaltwins.azure.net`) is the same value stored in the project's .env file and used by the Python execution engine for authentication through the `azure-digitaltwins-core` SDK.

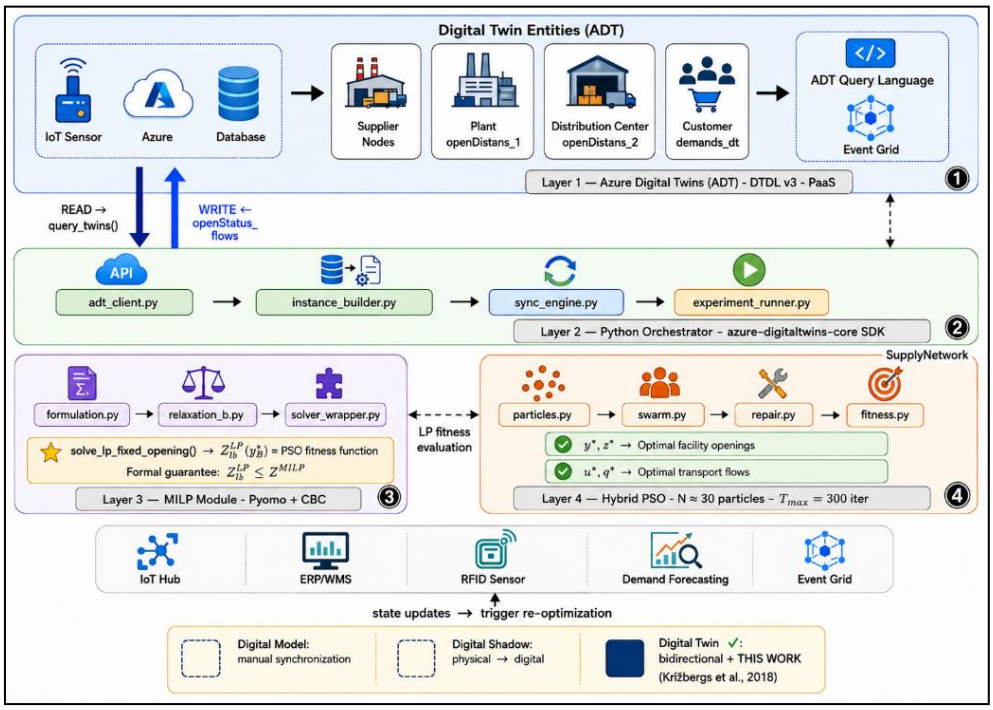


**Figure 1.** Four-layer cloud-based architecture of the proposed system: Azure Digital Twins (Layer 1), Python execution engine (Layer 2), MILP–Pyomo module (Layer 3), and hybrid PSO (Layer 4), with bidirectional synchronization

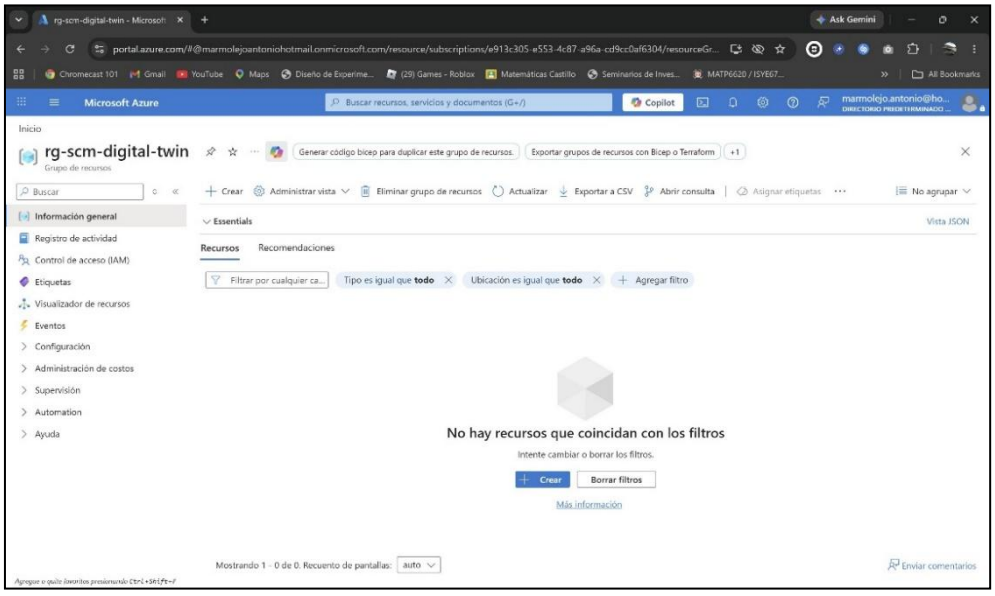


**Figure 2.** Azure Digital Twins instance deployed in the Azure Portal, showing the hostname (adt-scm-suministro.api.eus.digitaltwins.azure.net), the East US region, and an **Active** status, thereby confirming the availability of the cloud service for integration with the Python orchestrator

### 3.2.2. Layer 1: Ontological Modeling in Azure Digital Twins

The ADT instance manages a graph comprising four types of digital twins modeled using DTDL v3 (Microsoft, 2023): *Supplier*, *Plant*, *DC*, and *Customer*. The complete ontology, including its DTDL properties and relationships, is illustrated in Figure 3. Integration with Azure IoT Hub and Event Grid enables

automatic synchronization (Marmolejo-Saucedo, 2022; Milenkovic, 2022; Somma et al., 2025). The *openStatus* property in the *Plant* and *DC* models directly corresponds to the binary decision variables (y_i) and (z_j) of the MILP model. The instance graph for a specific demonstration network is presented in Figure 6.

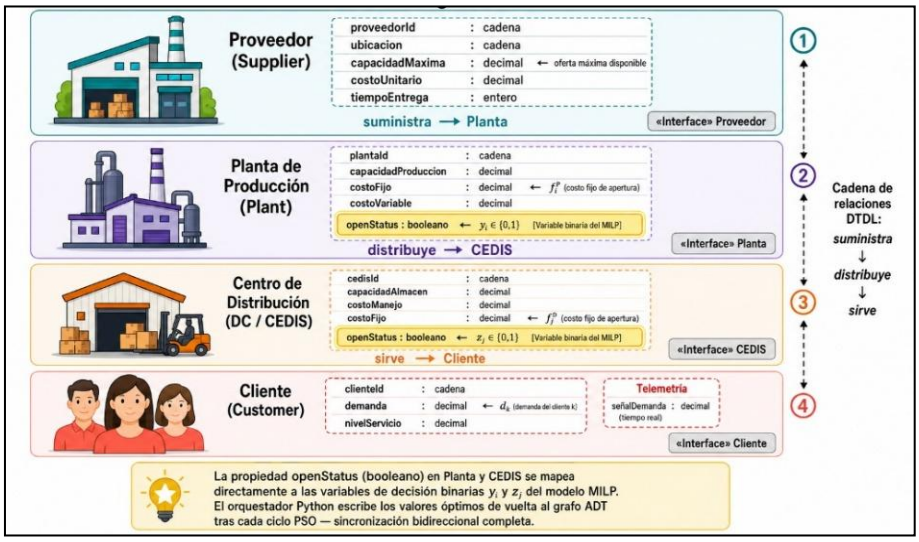


**Figure 3.** DTDL v3 ontology for a multi-echelon supply chain in Azure Digital Twins

Once the instance is active, the next step is to upload the four DTDL models using create_models.py. Figure 4 shows these models after they have been successfully registered and made available in Azure Digital Twins Explorer. Figure 5 presents the internal structure of the Customer model in JSON-LD format. The declared properties, particularly demand (schema: double), correspond exactly to those queried by instance_builder.py when constructing the demand vector for the MILP model.

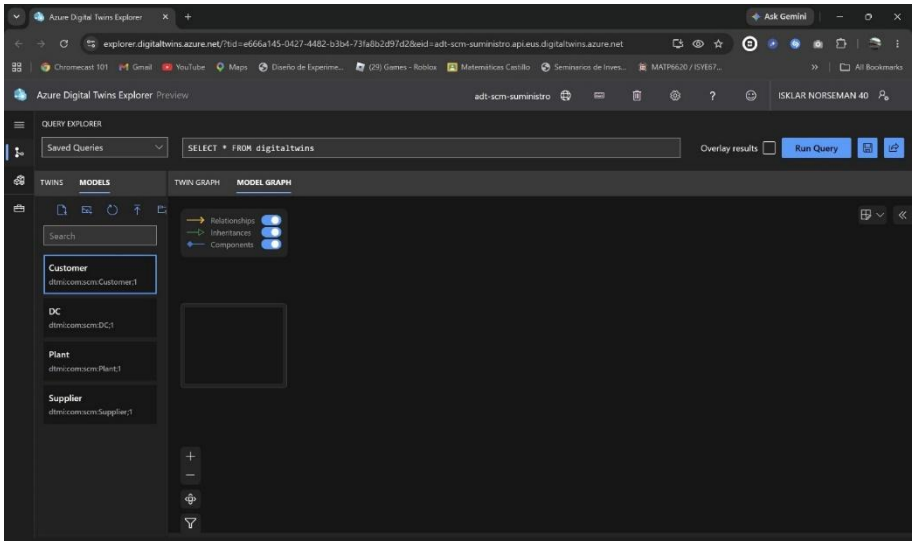


**Figure 4.** Supply chain DTDL models uploaded to Azure Digital Twins Explorer, showing their complete identifiers (dtmi:com:scm:Customer;1, dtmi:com:scm:DC;1, dtmi:com:scm:Plant;1, and dtmi:com:scm:Supplier;1)

The DTDL definition of the Customer model shown in Figure 5 follows the JSON-LD standard established by Microsoft (2023). Each property specifies its data type and schema, enabling the Python execution engine to read the values directly from the ADT graph without requiring an additional communication layer. The Telemetry field included in the model also enables the future incorporation of real-time demand signals, representing a natural extension of the proposed system for future research.

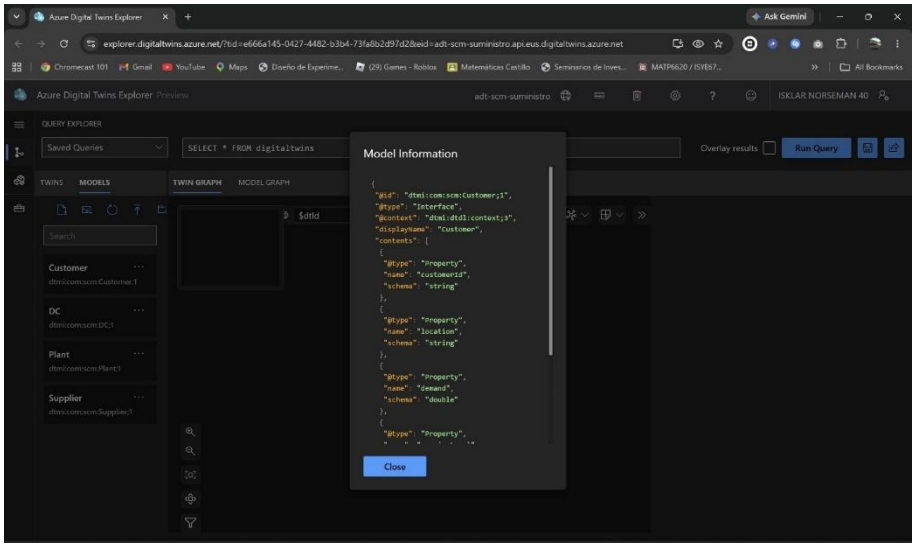


**Figure 5.** JSON-LD definition of the DTDL Customer model in Azure Digital Twins Explorer, including the customerId, demand, location, and serviceLevel properties. The demand field (schema: double) is read by instance_builder.py to construct the demand vector for the MILP model

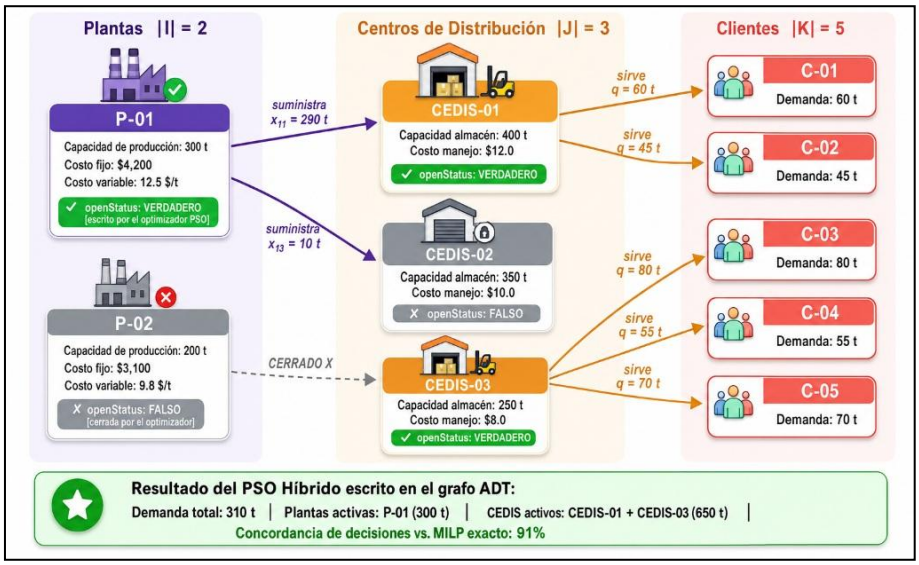


**Figure 6.** Instance graph in Azure Digital Twins: two plants, three distribution centers, and five customers, including DTDL properties, flow relationships (supplies, distributes, and serves), and the openStatus values written by the hybrid PSO after the optimization cycle

### 3.2.3. Layer 2: Python Execution Engine (Orchestrator) and ADT Interface

The interface is implemented using the azure-digitaltwins-core SDK (Microsoft, 2025a). Table 1 describes the main components. The complete bidirectional synchronization protocol is illustrated in Figure 7.

**Table 1.** ADT–Python integration components and synchronization protocol

| Component | Python Module | ADT Operation | Description |
|---|---|---|---|
| Authentication | adt_client.py | DefaultAzureCredential | Credentials are managed through a Service Principal or Managed Identity, with no keys stored in the source code. |
| State retrieval | sync_engine.py | query_twins() / get_digital_twin() | Uses Azure Digital Twins Query Language to retrieve the current capacities, costs, and demand values of all nodes in the graph. |
| Model construction | instance_builder.py | (local) | Maps the ADT graph to a SupplyNetwork object and subsequently to a Pyomo ConcreteModel, using dynamically updated parameters. |

| | | | |
|---|---|---|---|
| PSO execution | pso_hibrido.py | (local) | The hybrid PSO runs in local memory and calls solve_lp_fixed_openings() as the fitness function during each evaluation. |
| Decision write-back | sync_engine.py | update_digital_twin() | Writes the openStatus property (bool) to the Plant and DC twins and stores the optimal flows in the DTDL relationships, thereby |
| Reoptimization trigger | adt_client.py | Event Grid subscription | Subscribes to change events in ADT. If demand or capacity changes by more than 5%, a new PSO cycle is automatically triggered |

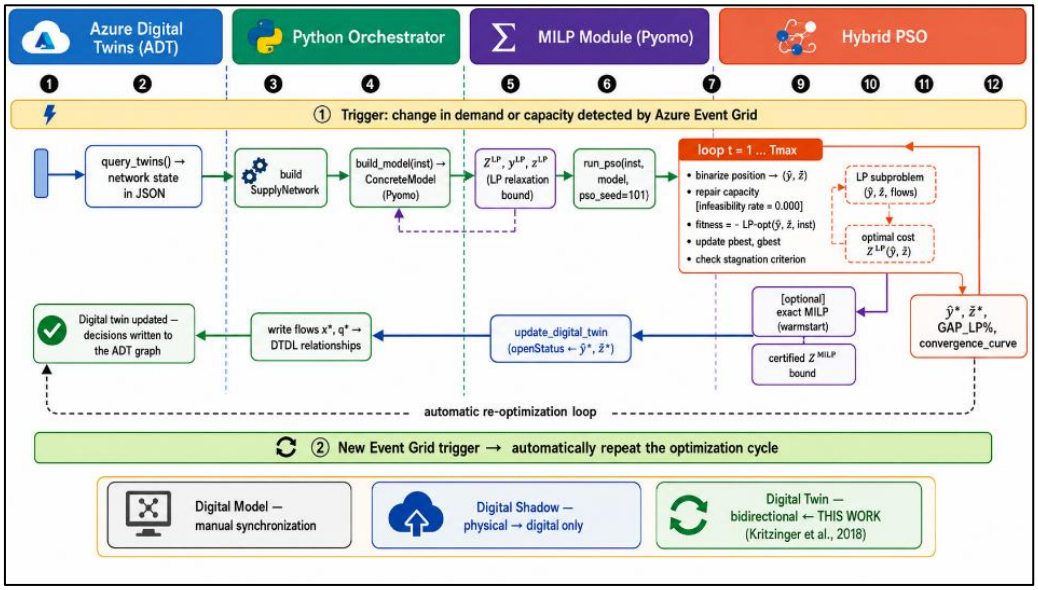


**Figure 7.** Bidirectional synchronization protocol between Azure Digital Twins and the PSO–MILP optimization engine: UML sequence diagram showing the 12 interactions in the cycle, from its initiation by Event Grid to the writing of openStatus values to the ADT graph

Figure 8 presents the complete system. The instance graph in Azure Digital Twins Explorer contains 10 digital twins connected by arrows representing the DTDL relationships. In the panel on the right side of the image, the DC-02 instance has an openStatus value of false. This value is calculated by the hybrid PSO based on the LP relaxation of the MILP model and written back to the digital twin using update_digital_twin(). This property demonstrates the bidirectional synchronization that distinguishes a digital twin from a unidirectional monitoring system (Kritzinger et al., 2018).

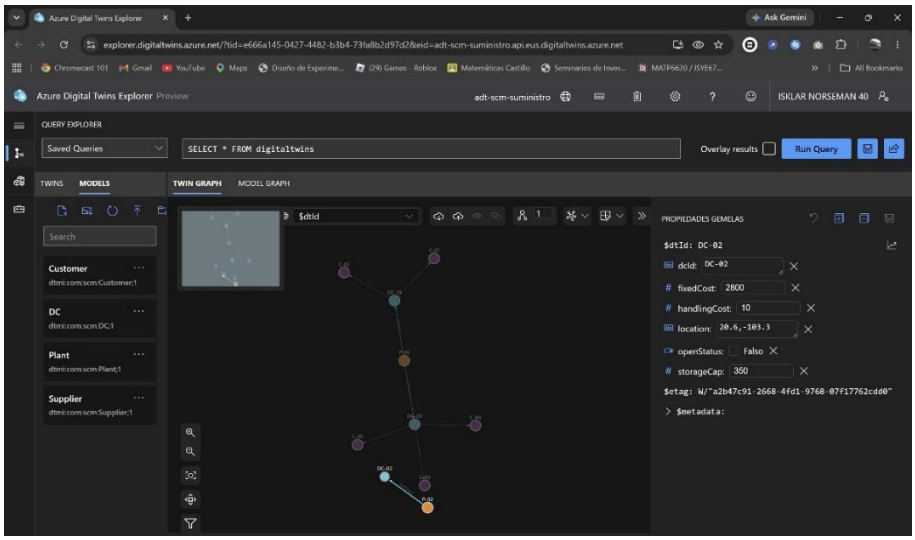


**Figura 8.** Grafo de instancias en Azure Digital Twins Explorer: 10 gemelos conectados con relaciones DTDL

The Anaconda Prompt terminal shown in Figure 9 validates the optimization process cycle. Step 1 confirms that the execution engine retrieved the 10 digital twins from ADT, with a total demand of 310 units. Step 2 presents the PSO results, yielding an optimal cost of $199,572.85 after 40 iterations, with an infeasibility rate of 0.0000. This confirms that the repair operator functioned without failure throughout all swarm evaluations. Step 3 lists the information sent to the ADT graph, thereby closing the loop between the optimizer and the digital twin.

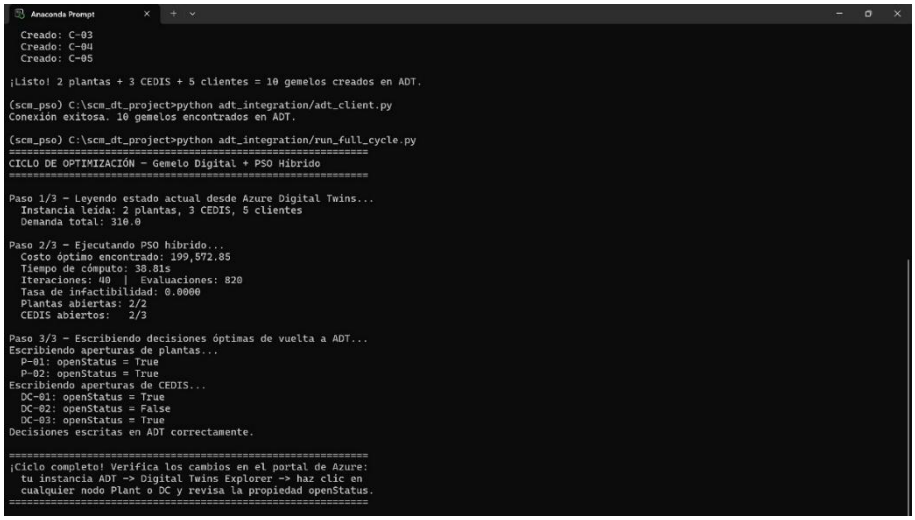


**Figure 9.** Execution of the complete cycle in Anaconda Prompt: Step 1, data retrieval from ADT (total demand of 310 units); Step 2, hybrid PSO (cost of $199,572.85, infeasibility rate of 0.0000, and 40 iterations); Step 3, writing of openStatus values to the ADT graph (DC-02: False; all remaining instances: True)

The MILP module and the PSO have no direct dependencies on the Azure API, which is a defining feature of the proposed digital twin design. These modules receive and return standard Python objects. This independence ensures experimental reproducibility in environments without ADT connectivity and facilitates system portability to other digital twin platforms, such as Eclipse Ditto or FIWARE (Somma et al., 2025). In addition, the standardization of data formats and communication protocols is essential to ensure adequate interoperability among platforms and supply chain stakeholders (Wasi et al., 2025).

### 3.2.4. Layer 4: Hybrid PSO Algorithm

*Particle Encoding*

Each particle encodes the facility-opening variables as a real-valued vector $p_n^t \in \mathbb{R}^{(|I|+|J|)}$ , following the formulation proposed by Kennedy and Eberhart (1997). Binarization is performed using a sigmoid function with a threshold of θ=0.5: $\hat{y}_i = 1$ if $\sigma(p_i) \geq \theta$, where $\sigma(p) = 1/(1+e^{-p})$. The dimensionality of the search space is reduced by solving the LP subproblem with fixed facility-opening decisions. Specifically, the number of heuristic variables decreases from |I||J|+|J||K|+|I|+|J| a solo |I|+|J|. The optimal flows $x_{ij}$ and $q_{jk}$ are obtained by solving this LP subproblem. The fitness function combines the fixed costs associated with the proposed facility-opening decisions and the optimal LP flow cost: $f(p) = \Sigma_i f_i^P \cdot \hat{y}_i + \Sigma_j f_j^D \cdot \hat{z}_j$ + LP-opt($\hat{y}, \hat{z}$). The complete pseudocode is presented in Figure 10.

$$\sigma(p) = \frac{1}{1+e^{-p}} \quad ; \quad f(p) = \sum_i^{\square} f_i^P \cdot \hat{y}_i + \sum_j^{\square} f_j^D \cdot \hat{z}_j + \text{LP-opt}(\hat{y}, \hat{z})$$

*Feasibility Repair Operator*

Before evaluating each particle, the algorithm verifies whether the installed capacity is sufficient to satisfy total demand. When capacity is insufficient, the repair operator opens additional facilities in descending order of capacity until the deficit is eliminated. This simple mechanism achieved an infeasibility rate of 0.000 across the 74 experimental runs conducted in this study.

*Hybridization through LP-Based Initialization*

The PSO update parameters follow the Clerc–Kennedy configuration: ω=0.729, $c_1$=$c_2$=1.494 (Shi & Eberhart, 1998). In the reference configuration, the LP-based initialization fraction is set to ρ_LP = 0.0, thereby ensuring statistical independence among the experimental runs.

$$v_n^{t+1}=\omega\cdot v_n^t+c_1\cdot r_1\cdot(\text{pbest}_n-p_n^t)+c_2\cdot r_2\cdot(\text{gbest}-p_n^t)$$

### 3.3. Implementation

The system was developed in Python 3.11. The MILP model was implemented using Pyomo 6.7+ (Hart et al., 2011), with CBC 2.10+ as the primary solver and GLPK 5.0+ for the LP subproblems. The ADT interface runs on azure-digitaltwins-core 1.2+ (Microsoft, 2025a). The PSO engine relies on NumPy 1.26+, while result processing is performed using pandas 2.0+ and figures are generated with matplotlib 3.8+.

The digital twin system is organized into seven modules: data_generation.py, milp_model.py, lp_relaxation.py, pso_hibrido.py, experiments.py, plots.py, and main.py, allowing each module to be tested independently. The experiments were conducted using strictly independent instance and PSO seeds, with seeds 1–10 assigned to the instances and seeds 101–110 assigned to the PSO runs. This configuration ensures the statistical independence of the experimental runs.

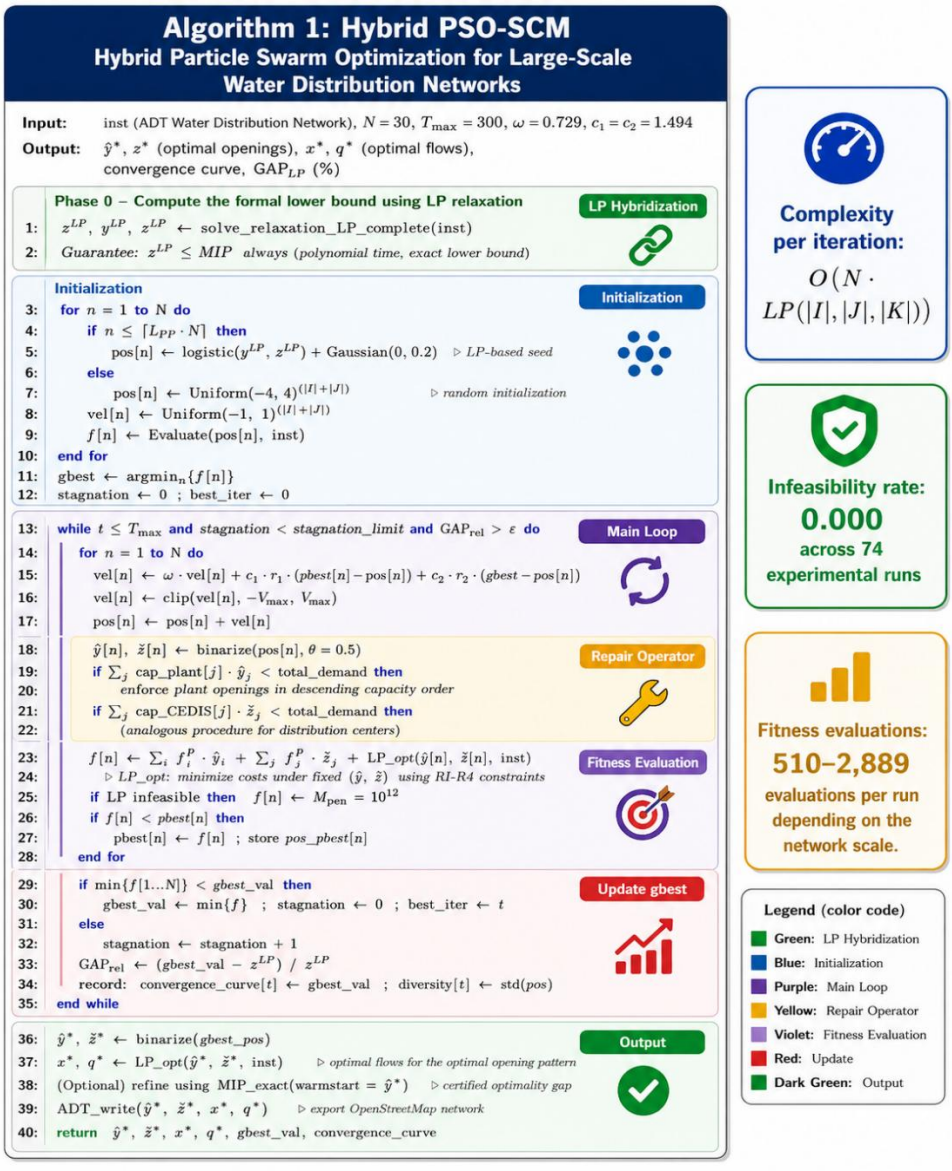


**Figure 10.** Pseudocode of Algorithm 1 (PSO-Hybrid-SCM) for large-scale supply network design

Figure 11 presents an independent verification of the ADT graph state, queried from Python directly through the Azure Digital Twins API. The results are consistent with the PSO solution: P-01, P-02, DC-01, and DC-03 appear as True (open), whereas DC-02 appears as False (closed by the optimizer). The five customers appear as N/A because the openStatus property does not apply to the Customer model.

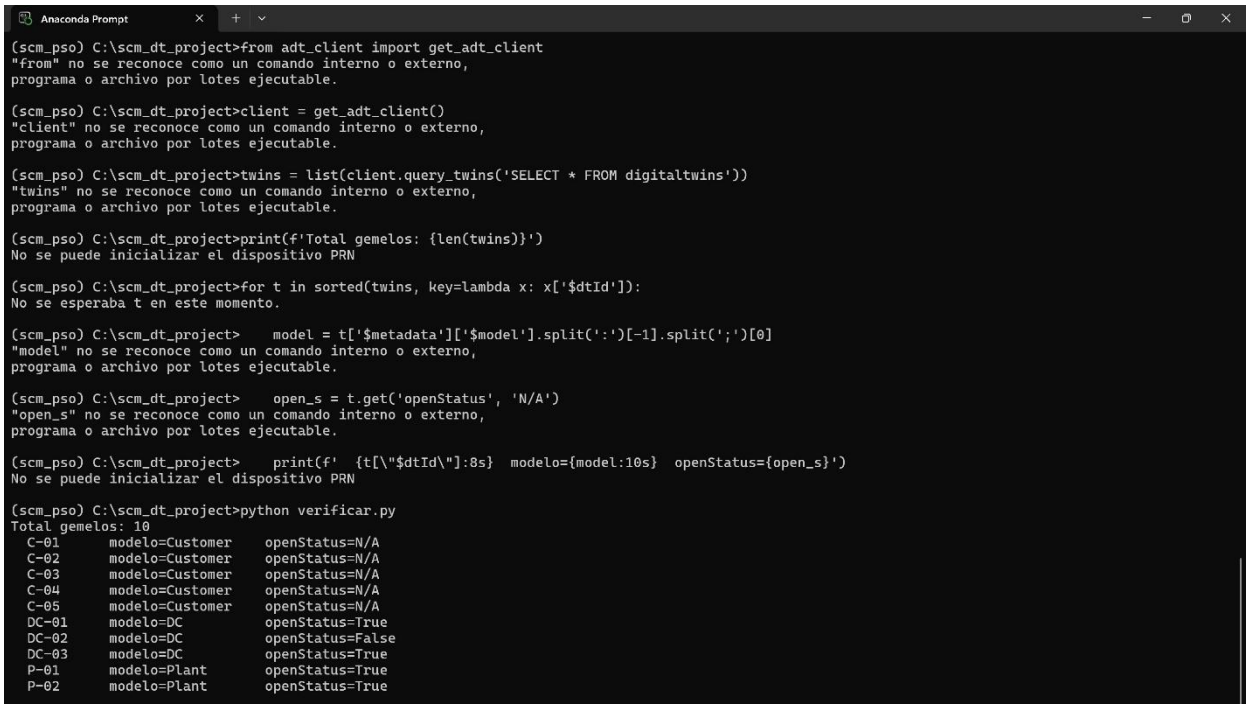


**Figure 11**. Verification of the ADT graph state after optimization through a direct query to the Azure Digital Twins API from Python

### 3.4. Experimental Desig

#### 3.4.1. Test Instances

Reproducible synthetic instances were generated at three scales: S (5 plants, 10 distribution centers, and 30 customers), M (10 plants, 20 distribution centers, and 80 customers), and XL (30 plants, 60 distribution centers, and 400 customers). Table 2 summarizes their dimensional structure. Compared with scale M, scale XL represents a threefold increase in the number of binary variables and an approximately fourteenfold increase in the number of continuous variables, placing the problem within a qualitatively different complexity regime.

**Table 2**. Structure of the test instances by scale.

| Escale | \|I\| | \|J\| | \|K\| | Binary Variables | Continuous Variables | Total Variables | Constraints | Integrality Density |
|---|---|---|---|---|---|---|---|---|
| S | 5 | 10 | 30 | 15 | 350 | 365 | 55 | 0.041 |
| M | 10 | 20 | 80 | 30 | 1,800 | 1,830 | 130 | 0.016 |
| XL | 30 | 60 | 400 | 90 | 25,800 | 25,890 | 550 | 0.003 |

For each scale, five complexity types were evaluated: standard; tight capacity, with capacity_ratio = 1.05; high fixed costs, generated using a Pareto distribution; asymmetric costs, generated by applying multiplicative noise to transportation costs; and a combined hard case, incorporating tight capacity, asymmetric costs, and clustered customers. Demand follows a log-normal distribution with a coefficient of variation of (CV = 0.3). The code is fully reproducible using fixed random seeds.

#### 3.4.2. Algorithm Configurations

The CBC solver was configured with ratioGap = 0.0001 and a time limit of 300 s for the S and M scales and 120 s for the XL scale. The hybrid PSO was configured with 30 particles, reduced to 20 for the XL scale; a maximum of 300 iterations, reduced to 50 for XL because of the evaluation limit; $\omega$=0.729; $c_1$=$c_2$=1.494;

stagnation_limit=60, and purely random initialization (ρ_LP=0.0). Nine independent runs were performed for each configuration at the S and M scales, corresponding to three instances and three PSO seeds, whereas four runs were conducted at the XL scale, corresponding to two instances and two PSO seeds. This resulted in a total of 74 experimental runs.

$$v_n^{t+1}=\omega\cdot v_n^t+c_1\cdot r_1\cdot\left(\text{pbest}_n-p_n^t\right)+c_2\cdot r_2\cdot(\text{gbest}-p_n^t)$$

### 3.4.3. Recorded Metrics

The metrics recorded for each run were: GAP_LP (the gap between the PSO solution and the LP bound), GAP_MIP (the gap between the PSO solution and the MILP optimum), MILP and PSO computation times, the number of iterations and fitness-function evaluations, the swarm infeasibility rate, final swarm diversity, consistency between the PSO and MILP facility-opening decisions for plants and distribution centers, and structural metrics of the problem instance.

The experiments were conducted on an ASUS Zenbook Duo UX8406MA equipped with an Intel Core Ultra 9 185H processor (2.50 GHz, x86-64 architecture) and 32 GB of RAM, running Windows 11 Home Single Language, version 25H2, build 26200.8737. The software environment consisted of Python 3.11, CBC 2.10, and GLPK 5.0.

## 4. RESULTS

### 4.1. Effectiveness of the Repair Operator

The PSO infeasibility rate was zero across all 74 experimental runs, for every scale and every type of instance evaluated. This result confirms that the capacity-based repair operator guarantees the feasibility of all swarm evaluations, even under highly heterogeneous problem structures.

### 4.2. Solution Quality and Integrality-Gap Pattern

Table 3 presents the comparative results regarding solution quality. Figure 12 complements the table with a boxplot visualization of the gap between the PSO solution and the LP bound by scale and instance type, revealing the relative dispersion among configurations with different levels of complexity. The central finding is the inverse correlation between the MILP problem's integrality gap and PSO solution quality, which was observed across all runs and at all three scales.

**Table 3.** Comparative solution-quality results (mean ± standard deviation). Green cells indicate a PSO/MILP gap below 3.5%

| Escale | Instance Type | MILP GAP LP (%) | GAP PSO/LP (%) | GAP PSO/MILP (%) | Consist P (%) | Consist DC (%) | Infeasi-bility |
|---|---|---|---|---|---|---|---|
| S | Standar | 2.45 | 3.26 ± 1.48 | 0.79 ± 0.99 | 93.3 | 90.0 | 0.000 |
| M | Standar | 2.37 | 3.56 ± 2.02 | 1.16 ± 1.72 | 94.4 | 91.1 | 0.000 |
| M | Tight capacity | 0.70 | 0.70 ± 0.16 | **0.00 ± 0.00** | 100.0 | 100.0 | 0.000 |
| M | High fixed costs | 2.92 | 5.70 ± 0.99 | 2.70 ± 1.09 | 82.2 | 82.2 | 0.000 |
| M | Asymmetric costs | 2.12 | 4.01 ± 2.23 | 1.85 ± 1.84 | 84.4 | 88.9 | 0.000 |
| M | Combined hard | 0.40 | 0.65 ± 0.39 | **0.25 ± 0.41** | 100.0 | 96.7 | 0.000 |
| XL | Standar | 1.53 | 13.56 ± 3.75 | 11.85 ± 3.86 | 70.8 | 67.9 | 0.000 |
| XL | Tight capacity | 0.28 | 2.45 ± 1.24 | **2.16 ± 1.26** | 91.7 | 92.1 | 0.000 |
| XL | High fixed costs | 1.01 | 8.66 ± 0.78 | 7.57 ± 0.81 | 80.0 | 70.4 | 0.000 |
| XL | Asymmetric costs | 1.33 | 14.41 ± 3.77 | 12.91 ± 3.83 | 76.7 | 67.9 | 0.000 |
| XL | Combined hard | 0.25 | 3.40 ± 0.91 | **3.15 ± 0.92** | 92.5 | 90.8 | 0.000 |

For instances with a low integrality gap—tight capacity at the M scale (0.70%) and combined hard at the M scale (0.40%)—the PSO reached the exact optimum, with mean gaps of 0.00% and 0.25%, respectively, and 100% consistency in facility-opening decisions. At the XL scale, under the same configurations—tight capacity (0.28%) and combined hard (0.25%)—the gap increased to 2.16% and 3.15%, respectively, with

decision consistency ranging from 91% to 93%. This result is consistent with the lower fitness-evaluation limit used at the XL scale: 510 evaluations, compared with 1,592–2,889 evaluations at the M scale. For instances with greater combinatorial complexity, such as the standard and asymmetric configurations at the XL scale, the gap exceeded 11%, indicating that 510 fitness evaluations are insufficient to adequately explore a 90-variable binary search space.

Figure 12 presents this distribution through boxplots grouped by scale and complexity type. The combined_hard and tight_capacity instances exhibit the lowest dispersion and the smallest gap values at both scales, whereas high_fixed_costs shows the greatest variability at the M scale. Figure 13 complements this analysis by presenting a heatmap for the M scale, which separates the effect of instance structure on the y-axis from the effect of algorithmic randomness on the x-axis. For example, for Instance 2 (Inst-2), the gap remains stable at 1.8% regardless of the PSO seed, whereas Instances 1 and 3 show greater variability with PSO-101, suggesting sensitivity to initialization for those specific problem structures.

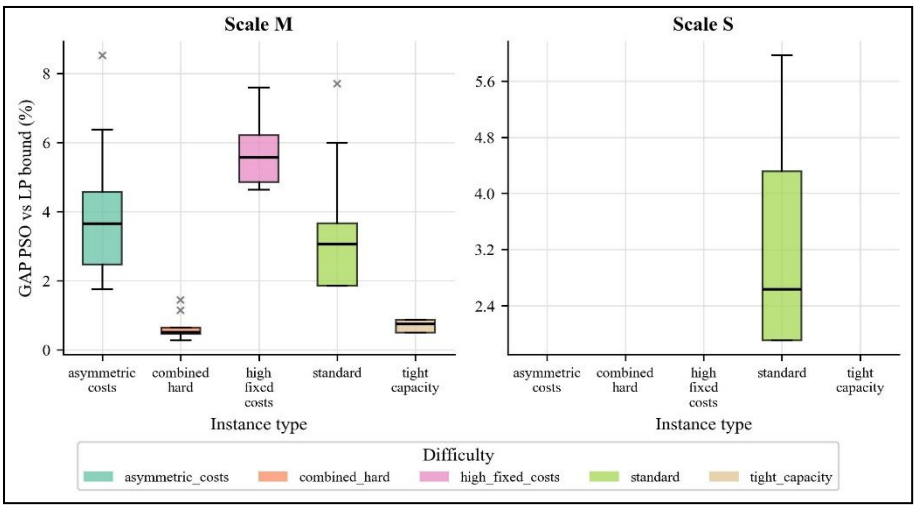


**Figure 12.** Distribution of the gap between the PSO solution and the LP bound by instance type at the S and M scales (nine runs per type). The combined_hard and tight_capacity instances exhibit the smallest gaps, consistent with their low integrality gaps

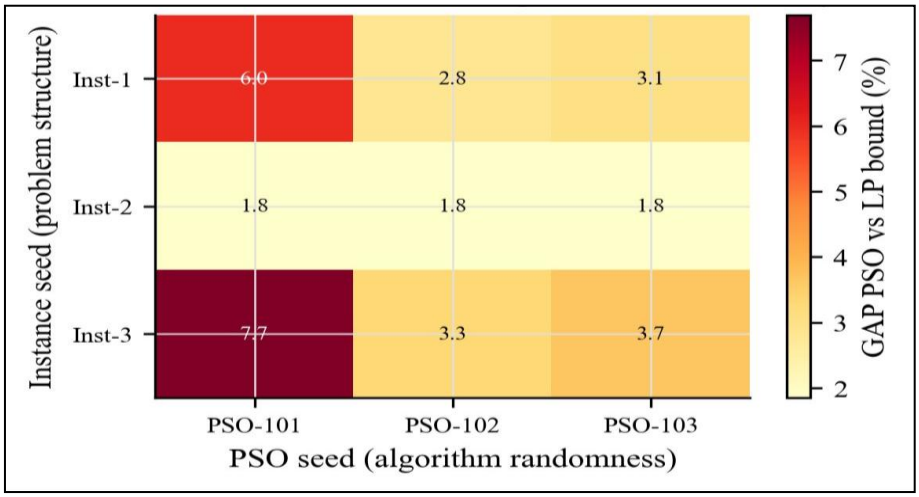


**Figure 13.** Heatmap of the gap between the PSO solution and the LP bound (%) for standard M-scale instances. The vertical axis reflects the effect of the problem structure, whereas the horizontal axis represents algorithmic randomness (PSO seed)

### 4.3. Computational Efficiency.

Table 4 presents the computation times by scale and instance type for all experiments. Figures 14 and 15 provide a visual comparison of computation times using a logarithmic scale for the S/M

and XL scales, respectively, thereby allowing the differences between the MILP and PSO approaches to be identified.

**Table 4**. Computation times: exact MILP versus hybrid PSO. Yellow cells indicate a PSO/MILP time ratio below 30x

| Scale | Instance Type | MILP t̄ (s) | MILP t_max (s) | PSO t̄ (s) | PSO t_max (s) | Ratio PSO/MILP |
|---|---|---|---|---|---|---|
| S | Standar | 0.14 | 0.21 | 92.2 | 138.5 | 646× |
| M | Standar | 0.41 | 0.65 | 328.0 | 492.4 | 811× |
| M | Combined hard | 0.19 | 0.36 | 138.6 | 169.1 | 734× |
| XL | Standar | 15.6 | 25.1 | 432.8 | 436.3 | **28×** |
| XL | Tight capacity | 2.1 | 2.4 | 500.1 | 556.4 | 234× |
| XL | High fixed costs | 20.0 | 24.6 | 546.4 | 561.3 | **27×** |
| XL | Asymmetric costs | 3.8 | 4.0 | 564.0 | 566.2 | 148× |
| XL | Combined hard | 2.3 | 2.6 | 612.1 | 642.6 | 264× |

For the S & M scales, Figure 14 shows that the CBC solver is 646–811 times faster than the PSO. However, Figure 15 reveals that, at the XL scale, this ratio decreases to ~60x, with a mean MILP computation time of 8.8 s compared with 531 s for the PSO. This behavior indicates a transition region in which the metaheuristic begins to become computationally competitive. This suggest that the proposed approach has the potential to scale effectively, as the computation-time ratio decreases with problem size, from (811x) at the M scale to less than (30x) for the most challenging XL instances for the exact solver.

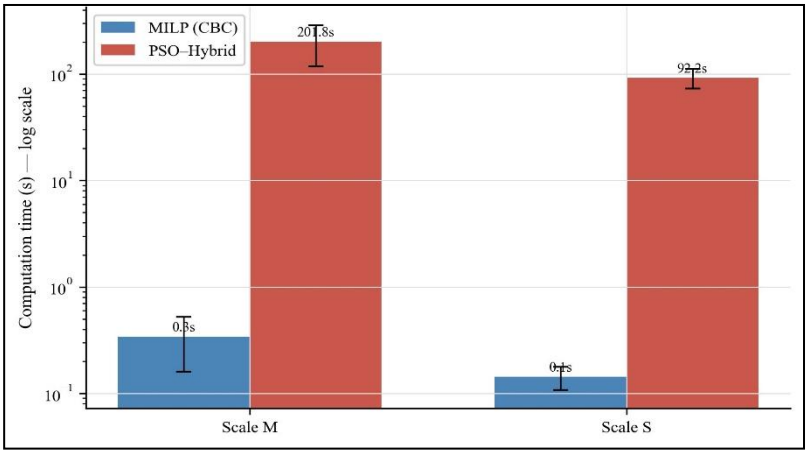


**Figure 14**. Comparison of MILP (CBC) and hybrid PSO computation times on a logarithmic scale for the S and M scales. CBC outperforms the PSO, with time ratios of (646\times) and (811\times), respectively

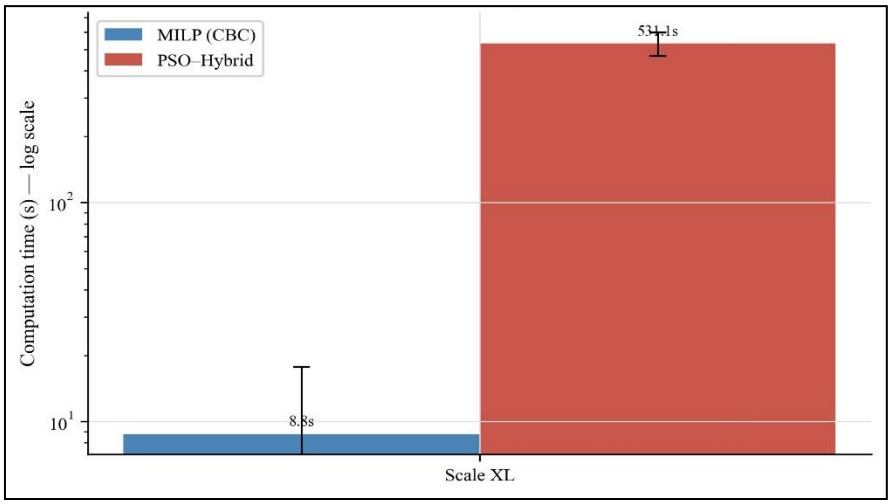


**Figure 15**. Comparison of MILP and hybrid PSO computation times for the XL scale. The PSO/MILP time ratio decreases to ∼60x

### 4.4. Swarm Behavior.

Figure 16 shows the normalized convergence curves, expressed as the gap relative to the LP bound, for the S and M scales. At the S scale, convergence is rapid, and the mean gap stabilizes at approximately 3.5% within the first 15 iterations. At the M scale, convergence is slower, and the mean gap remains at approximately 40–45%, reflecting the inclusion of instances with different levels of complexity in the overall average. Configurations associated with larger gaps, particularly high_fixed_costs and asymmetric, increase the mean value. Figure 17 shows the convergence behavior at the XL scale. The mean gap decreases from approximately 52% at iteration 1 to approximately 43% at iteration 50, confirming that the swarm was still improving when the fitness-evaluation limit was reached.

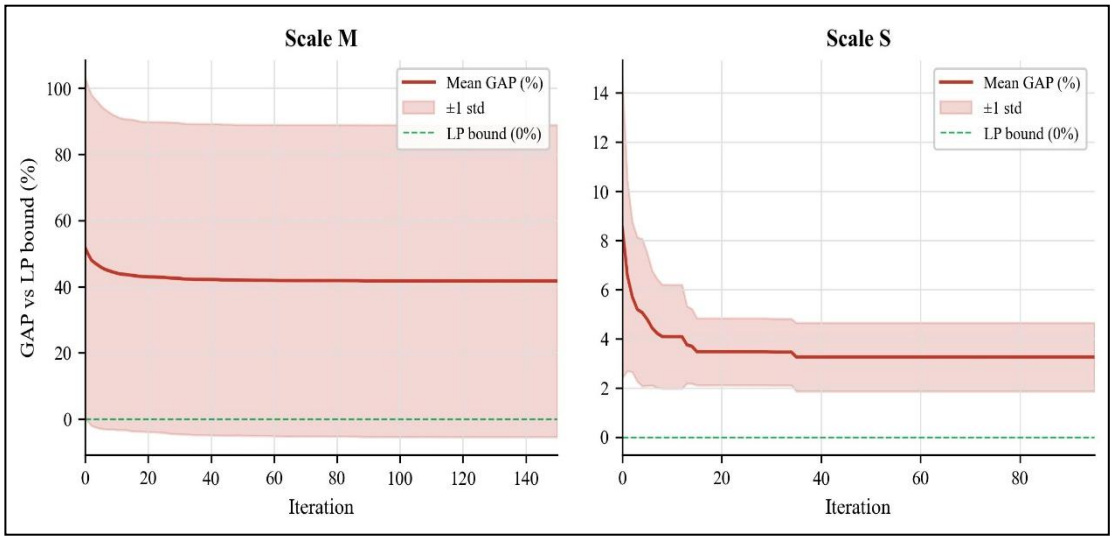


**Figure 16**. Convergence curves of the hybrid PSO for the S and M scales (mean ± 1 standard deviation). The dashed line indicates the LP bound (GAP = 0%). Convergence at the S scale is notably faster than at the M scale

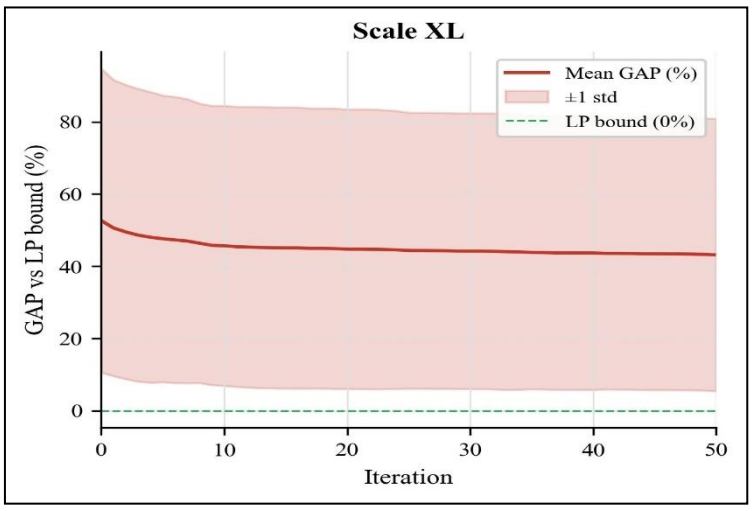


**Figure 17**. Convergence curves of the hybrid PSO for the XL scale (mean ± 1 standard deviation, 20 runs). The mean GAP decreases from approximately 52% to 43% over 50 iterations without reaching the stagnation criterion

### 4.5. Scalability Analysis: Integrated Comparison across the S, M, and XL Scales.

Figure 18 presents the boxplot of GAP values for the XL scale by instance type, complementing Table 3 by showing the complete distribution of results. The combined_hard and tight_capacity configurations exhibit the lowest GAP values, with median values of approximately 3.5–3.6%. This result is consistent with their low integrality gaps. In contrast, the asymmetric_costs and standard configurations show the highest median GAP values, at approximately 13–15%.

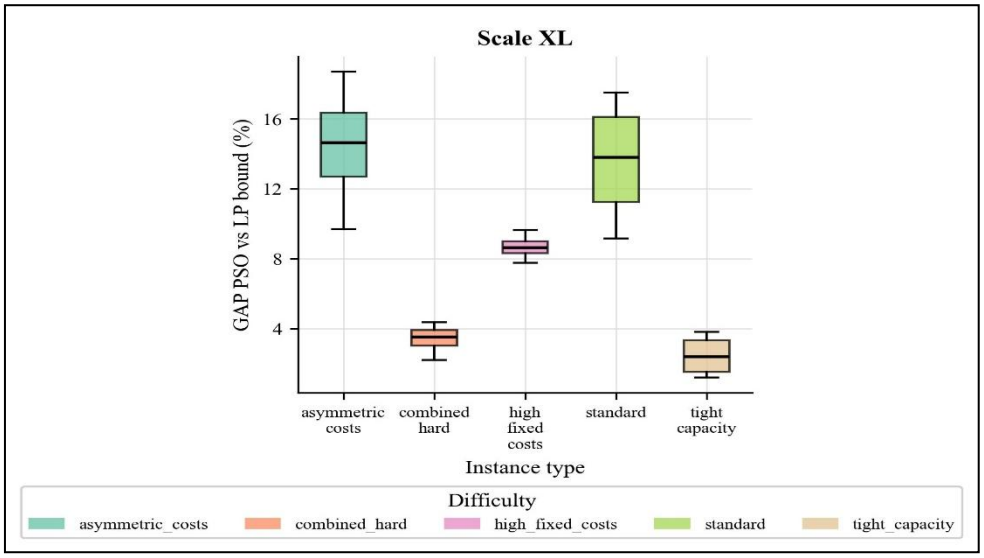


**Figure 18.** Distribution of the gap between the PSO solution and the LP bound by instance type at the XL scale (20 runs). The combined_hard and tight_capacity instance types exhibit the lowest median values (approximately 3.5%), whereas asymmetric_costs and standard show the highest median values (approximately 13–15%)

## 5. DISCUSSION

### 5.1. LP Hybridization as an Adaptive Mechanismo

The most relevant finding obtained from the experiments was the systematic correlation between the integrality gap and PSO solution quality. This relationship was observed across all 74 runs and over three orders of magnitude in problem size, as shown in Figure 12 for the S and M scales and Figure 18 for the XL scale. This correlation is a direct consequence of the hybridization mechanism. When the integrality gap is low, the LP signal discriminates with high resolution among binary configurations of different quality, allowing the swarm to converge rapidly (Figures 16 and 17). When the integrality gap is high, the LP signal becomes less effective.

These results support the proposed hybridization compared with approaches based solely on heuristic fitness functions (Noel & Jannett, 2004; Tirkolaee et al., 2020). The practical implication for the digital twin is that the system can calculate the integrality gap before invoking the PSO and use it as a predictor of expected solution quality, thereby guiding the allocation of the fitness-evaluation budget.

### 5.2. Topological Quality: The Digital Twin Reflects the Optimal Network.

Consistency in facility-opening decisions exceeds 90% even at the XL scale for instances with low integrality gaps, as shown in Table 3. This means that the openStatus property written to the ADT graph (Figure 3) is topologically correct for more than 90% of the network nodes, ensuring that the decision state represented by the digital twin is reliable for subsequent operations. This result validates the documented benefits of digital twins for supply chain resilience (Ivanov & Dolgui, 2021; Longo et al., 2023; Marmolejo-Saucedo, 2020). However, for more complex instances, such as XL-standard and XL-asymmetric, agreement decreases to 68–77%, highlighting the importance of an exact MILP refinement mechanism as a future extension of this work.

Figure 19 shows the architecture of the digital twin graph. It displays the properties of the P-01 twin in Azure Digital Twins Explorer, with openStatus set to True, together with the parameters retrieved by the execution engine at the beginning of the cycle: fixedCost: $4,200, productionCap: 300 t, and variableCost: $12.5/t. The openStatus value was calculated by the PSO, sent to the Azure API through a JSON Patch operation, and subsequently displayed in the graph. Thus, information flows from the optimizer back to the digital twin. This bidirectional exchange places the implementation within the Digital Twin category of the

taxonomy proposed by Kritzinger et al. (2018), rather than the Digital Shadow category, in which data flow is unidirectional.

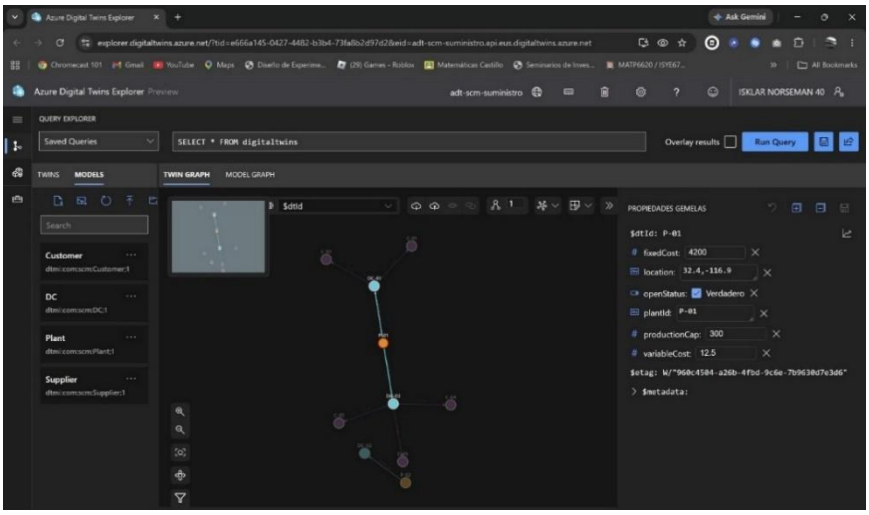


**Figura 19.** Propiedades del gemelo P-01 en Azure Digital Twins Explorer: openStatus = Verdadero escrito por el PSO híbrido mediante update_digital_twin(), junto con fixedCost ($4,200), productionCap (300 t) y variableCost ($12.5/t) leídos desde el grafo ADT al inicio del ciclo

### 5.3. Scalability and the Competitive Transition Region

Figures 14 and 15 show that the PSO/MILP computation-time ratio, calculated from the mean times aggregated by scale, decreases from (811\times) for the M-standard configuration to approximately (60\times) at the XL scale. This reduction characterizes a transition region in which the metaheuristic begins to become computationally competitive. However, the literature reports that instances with more than 200 binary variables may require several hours of computation without guaranteeing optimality (Avella et al., 2023; Melo et al., 2009). Previous experience with large-scale digital twins (Marmolejo-Saucedo, 2022) suggests that XXL-scale instances in real industrial environments would necessarily require the PSO–ADT combination to maintain operationally acceptable response times.

### 5.4. Swarm Behavior and Optimal Evaluation Budget

Figure 17 shows that, at the XL scale, the swarm was still improving when the 50-iteration limit was reached, with a mean best_iter value of 44.8 for the XL-standard configuration. In contrast, convergence occurred early for XL-combined_hard, with best_iter = 12.0. The initial diversity of the XL swarm, exceeding 20 at iteration 0, is considerably greater than that observed at the M scale (approximately 12) and the S scale (approximately 8), reflecting the larger search space. This behavior suggests that the recommended limit for the XL scale should be at least 150 iterations with 30 particles, equivalent to the configuration used at the M scale. Such a configuration could produce GAP values comparable to those observed at the M scale for the same instance types.

### 5.5. Limitations

At all evaluated scales, CBC certified optimality within 0.14–20.0 s, a condition under which the PSO provides no advantage in terms of computation time. The XL configuration, limited to 510 fitness evaluations, reached the maximum number of iterations without satisfying the stagnation-based convergence criterion for instances with larger integrality gaps. The potential effect of communication latency at an industrial scale was not considered.

Furthermore, validation of the ADT integration was conducted only on reduced-scale demonstration instances. All test instances were synthetic; therefore, validation using real-world data from industrial supply chains remains an important direction for future research.

## FUNDING

The authors received no financial support for the research, authorship, or publication of this article.

## CONFLICTS OF INTERES

The authors declare that there are no conflicts of interest related to the subject matter of this study.

## AUTHOR CONTRIBUTIONS

Conceptualization: Marmolejo-Saucedo, J. A.; Rodríguez-Aguilar, R.; Mendoza, A. Data curation: Rodríguez-Aguilar, R.; Mendoza, A. Formal analysis: Marmolejo-Saucedo, J. A.; Rodríguez-Aguilar, R. Investigation: Marmolejo-Saucedo, J. A.; Rodríguez-Aguilar, R.; Mendoza, A. Methodology: Marmolejo-Saucedo, J. A.; Rodríguez-Aguilar, R.; Mendoza, A. Project administration: Marmolejo-Saucedo, J. A. Software: Rodríguez-Aguilar, R.; Mendoza, A. Validation: Marmolejo-Saucedo, J. A.; Mendoza, A. Visualization: Rodríguez-Aguilar, R.; Mendoza, A. Writing—original draft: Marmolejo-Saucedo, J. A.; Rodríguez-Aguilar, R.; Mendoza, A. Writing—review and editing: Marmolejo-Saucedo, J. A.; Rodríguez-Aguilar, R.; Mendoza, A.

## REFERENCES

Avella, P., Calamita, A., & Palagi, L. (2023). A computational study of off-the-shelf MINLP solvers on a benchmark set of congested capacitated facility location problems. *ArXiv*. https://doi.org/10.48550/arXiv.2303.04216

Dominguez Martinez, V., Parody De La Cruz, J. D., Guerra Manjarrez, J. F., & Rodriguez Velandia, D. A. (2026). *Digital twins in healthcare (2018–2024): A scientometric analysis with emphasis on cardiology and perspectives for Colombia*. En M. Zuluaga (Ed.), *Scientometric Perspectives on Biomedical Research: Applications Across Oncology, Infectious Diseases, Bioinformatics, and Medical Technologies* (Vol. 1, pp. 33-50). Contemporary Biomedical Research. Pro-Metrics. https://doi.org/10.47909/978-9916-9331-7-6.5

Farahani, R. Z., Rezapour, S., Drezner, T., & Fallah, S. (2014). Competitive supply chain network design: An overview of classifications, models, solution techniques and applications. *Omega*, *45*, 92–118. https://doi.org/10.1016/j.omega.2013.08.006

Geoffrion, A. M., & Graves, G. W. (1974). Multicommodity Distribution System Design by Benders Decomposition. *Management Science*, *20*(5), 822–844. https://doi.org/10.1287/mnsc.20.5.822

Grieves, M. (2014). *Digital Twin: Manufacturing Excellence through Virtual Factory Replication*.

Grieves, M., & Vickers, J. (2017). Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems. In F.-J. Kahlen, S. Flumerfelt, & A. Alves (Eds.), *Transdisciplinary Perspectives on Complex Systems: New Findings and Approaches* (pp. 85–113). Springer International Publishing. https://doi.org/10.1007/978-3-319-38756-7_4

Guo, D., & Mantravadi, S. (2025). The role of digital twins in lean supply chain management: review and research directions. *International Journal of Production Research*, *63*(5), 1851–1872. https://doi.org/10.1080/00207543.2024.2372655

Hart, W. E., Watson, J.-P., & Woodruff, D. L. (2011). Pyomo: Modeling and solving mathematical programs in Python. *Mathematical Programming Computation*, *3*(3), 219–260. https://doi.org/10.1007/s12532-011-0026-8

Ivanov, D., & Dolgui, A. (2019). The impact of digital technology and Industry 4.0 on the ripple effect and supply chain risk analytics. *International Journal of Production Research*, *57*(3), 829–846. https://doi.org/10.1080/00207543.2018.1488086

Ivanov, D., & Dolgui, A. (2021). A digital supply chain twin for managing the disruption risks and resilience in the era of Industry 4.0. *Production Planning & Control*, *32*(9), 775–788. https://doi.org/10.1080/09537287.2020.1768450

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks* (Vol. 4, pp. 1942–1948). IEEE. https://doi.org/10.1109/ICNN.1995.488968

Kennedy, J., & Eberhart, R. C. (1997). A discrete binary version of the particle swarm algorithm. In *1997 IEEE International Conference on Systems, Man, and Cybernetics: Computational Cybernetics and Simulation* (Vol. 5, pp. 4104–4108). IEEE. https://doi.org/10.1109/ICSMC.1997.637339

Kritzinger, W., Karner, M., Traar, G., Henjes, J., & Sihn, W. (2018). Digital Twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine*, *51*(11), 1016–1022. https://doi.org/10.1016/j.ifacol.2018.08.474

Lim, K. Y. H., Zheng, P., & Chen, C.-H. (2020). A state-of-the-art survey of Digital Twin: techniques, engineering product lifecycle management and business innovation perspectives. *Journal of Intelligent Manufacturing*, *31*(6), 1313–1337. https://doi.org/10.1007/s10845-019-01512-w

Longo, F., Mirabelli, G., Padovano, A., & Solina, V. (2023). The Digital Supply Chain Twin paradigm for enhancing resilience and sustainability against COVID-like crises. *Procedia Computer Science*, *217*, 1940–1947. https://doi.org/10.1016/j.procs.2022.12.394

Marmolejo-Saucedo, J. A. (2020). Design and Development of Digital Twins: A Case Study in Supply Chains. *Mobile Networks and Applications*, *25*(6), 2141–2160. https://doi.org/10.1007/s11036-020-01557-9

Marmolejo-Saucedo, J. A. (2022). Digital Twin Framework for Large-Scale Optimization Problems in Supply Chains: A Case of Packing Problem. *Mobile Networks and Applications*, *27*(5), 2198–2214. https://doi.org/10.1007/s11036-021-01856-9

Melo, M. T., Nickel, S., & Saldanha-da-Gama, F. (2009). Facility location and supply chain management - A review. *European Journal of Operational Research*, *196*(2), 401–412. https://doi.org/10.1016/j.ejor.2008.05.007

Microsoft. (2023). Digital Twins Definition Language (DTDL), Version 3. In *Azure Open Digital Twins*. https://azure.github.io/opendigitaltwins-dtdl/DTDL/v3/DTDL.v3.html

Microsoft. (2025a). Azure Digital Twins Core client library for Python (azure-digitaltwins-core). In *Python Package Index (PyPI)* (1.3.0). Python Software Foundation. https://pypi.org/project/azure-digitaltwins-core/

Microsoft. (2025b). What is an ontology in Azure Digital Twins? In *Microsoft Learn*. https://learn.microsoft.com/en-us/azure/digital-twins/concepts-ontologies

Microsoft. (2025c). What is Azure Digital Twins? In *Microsoft Learn*. https://learn.microsoft.com/en-us/azure/digital-twins/overview

Milenkovic, M. (2022). Internet of Things: System Reference Architecture. *ArXiv*. https://doi.org/10.48550/arXiv.2204.01872

Noel, M. M., & Jannett, T. C. (2004). Simulation of a new hybrid particle swarm optimization algorithm. In *Proceedings of the Thirty-Sixth Southeastern Symposium on System Theory* (pp. 150–153). IEEE. https://doi.org/10.1109/SSST.2004.1295638

Pérez-Baquero, R.-S., Murgas Gutierrez, J. P., Arzuaga Gomez, A. C., & Suarez Ravelo, C. D. (2026). Decision technologies in control engineering: A scientometric review of LQR control for inverted pendulum

systems (2006–2025). *DecisionTech Review*, *6*, 1–14. https://doi.org/10.47909/dtr.36

Shi, Y., & Eberhart, R. C. (1998). A modified particle swarm optimizer. In *1998 IEEE International Conference on Evolutionary Computation Proceedings; IEEE World Congress on Computational Intelligence* (pp. 69–73). IEEE. https://doi.org/10.1109/ICEC.1998.699146

Simchi-Levi, D., Kaminsky, P., & Simchi-Levi, E. (2008). *Designing and Managing the Supply Chain: Concepts, Strategies, and Case Studies* (3rd ed.). McGraw-Hill/Irwin.

Somma, A., Amalfitano, D., De Benedictis, A., & Pelliccione, P. (2025). TwinArch: A digital twin reference architecture. *Journal of Systems and Software*, 112613. https://doi.org/10.1016/j.jss.2025.112613

Tao, F., Zhang, H., Liu, A., & Nee, A. Y. C. (2019). Digital Twin in Industry: State-of-the-Art. *IEEE Transactions on Industrial Informatics*, *15*(4), 2405–2415. https://doi.org/10.1109/TII.2018.2873186

Tirkolaee, E. B., Goli, A., Faridnia, A., Soltani, M., & Weber, G.-W. (2020). Multi-objective optimization for the reliable pollution-routing problem with cross-dock selection using Pareto-based algorithms. *Journal of Cleaner Production*, *276*, 122927. https://doi.org/10.1016/j.jclepro.2020.122927

Wasi, A. T., Anik, M. A., Rahman, A., Hoque, M. I., Islam, M. D. S., & Ahsan, M. M. (2025). A Theoretical Framework for Graph-based Digital Twins for Supply Chain Management and Optimization. *ArXiv*. https://doi.org/10.48550/arXiv.2504.03692