

Business-Semantic-Aware Joint Orchestration of Data, Computing Services, and Resources for Smart Grid Data Fabrics

Yingcheng Gu¹, Shengjie Wei¹, Liu Wei¹, Fei Xia¹, Meizhao Liu¹

¹Information and Telecommunication Branch, State Grid Jiangsu Electric Power Co., Ltd., Jiangsu, China

Corresponding author: Yingcheng Gu (e-mail: 623792988@qq.com).

ABSTRACT Smart-grid operational analytics rely on heterogeneous data, analytical services, and distributed computing resources, but existing discovery and unified-access mechanisms do not ensure executable orchestration under operational constraints. This paper proposes BS-DCSO, a business-semantic-aware method for jointly selecting data services, computing services, and resource nodes in a smart-grid data-fabric environment. BS-DCSO integrates type- and permission-aware filtering, QoS-based top-k pruning, business-context scoring, complete-plan evaluation, feasibility-first lexicographic selection, and failure-triggered re-orchestration. Experiments on a synthetic benchmark covering five representative smart-grid scenarios and ten random seeds show a provisioning success rate of 0.4897 ± 0.1139 , a feasible-instance success rate of 0.8320 ± 0.0562 , and a constraint violation rate of 0.0809 ± 0.0213 . Compared with ResourceFirstScheduling, BS-DCSO improves paired provisioning success by 0.1423 ± 0.0552 , while outperforming five controlled baselines across the three main metrics. Re-orchestration recovers feasible replacements in 0.8282 ± 0.0398 of eligible single-service failures. Under the sequential fixed-catalogue timing protocol, the method requires 151.39–175.58 ms per request for 100–1000 requests, indicating a practical feasibility–computation trade-off under the tested synthetic setting rather than high-concurrency scalability.

INDEX TERMS smart grid, data fabric, service orchestration, operational analytics, business semantics, service composition, failure recovery.

I. INTRODUCTION

THE digitalization of modern power systems has created a large and heterogeneous operational-data environment spanning supervisory control and data acquisition systems, phasor measurement units, substation monitoring systems, assetmanagement platforms, alarm and maintenance records, meteorological services, and market-related information systems [1]–[3]. These data support smart-grid applications such as situational awareness, load forecasting, renewable-energy monitoring, fault analysis, equipment diagnosis, and operational decision support. In practical utility environments, however, the information required by one analytical task is often distributed across multiple platforms and exposed through heterogeneous data interfaces, service APIs, and computational tools [4], [5]. Data catalogues, metadata-management systems, data spaces, and virtualization technologies improve the discovery, interpretation, and unified access of heterogeneous energy data [6]–[8]. These mechanisms are essential for identifying relevant data assets, but an operational analytics request generally requires more

than data retrieval. A complete execution chain may require several data products, preprocessing or analytical services, and an appropriate edge or cloud resource node while simultaneously satisfying latency, data-quality, reliability, cost, authorization, and capacity constraints. Consequently, discovering suitable data does not by itself determine an executable provisioning plan.

This distinction exposes a cross-layer orchestration gap. Workflow scheduling and service-placement methods commonly optimize task-to-resource mapping when task structures or candidate services are already specified [9]–[11]. Semantic middleware and knowledge-graph-based approaches improve the interpretation and retrieval of heterogeneous grid information but do not necessarily determine a constraint-feasible composition and deployment of functional services [12]–[14]. Service-composition research addresses functional and QoS-aware service selection, but it is typically separated from smart-grid business-context interpretation and deployment-resource selection [15]. For a smart-grid data fabric, these decisions are coupled: the

business request determines the required data and analytical functions, candidate services determine achievable quality, reliability, latency, and cost, and the selected resource node determines whether the resulting composition can be executed under capacity and load constraints.

To address this problem, this paper proposes BS-DCSO, a Business-Semantic-Aware Data and Computing Service Orchestration method for smart-grid data fabrics. BS-DCSO first constructs exact-type candidate pools subject to permission availability, applies QoS-oriented top-k pruning, and then jointly evaluates retained data-service and computing-service combinations on all available resource nodes. A lightweight TF-IDF representation provides business-context relevance during complete-plan evaluation, while a feasibility-first lexicographic rule ensures that a constraint-feasible plan is preferred to an infeasible alternative regardless of its soft utility. In this paper, the term “business-semantic-aware” denotes lightweight lexical business-context differentiation rather than ontology-level semantic reasoning. When a selected functional service becomes unavailable, the failed service is excluded and the same procedure is executed on the reduced catalogue to search for a replacement plan.

The present study targets the ICT and data-provisioning layer that supports smart-grid operational analytics and decision-support applications. It does not claim timing guarantees for protective relaying, safety-critical control, or other hard real-time closed-loop functions. Within this scope, the main contributions are as follows:

- A joint smart-grid provisioning formulation couples business context and functional requirements with data quality, reliability, latency, cost, permission, and heterogeneous resource-capacity constraints, so that data-service selection, computing-service selection, and deployment-node selection are evaluated at complete-plan level.
- A bounded feasibility-oriented orchestration method combines per-type QoS pruning with Cartesian complete-plan construction, exhaustive resource-node evaluation within the retained service space, and feasibility-first lexicographic selection. This design explicitly separates hard executability from soft multi-attribute preference.
- A failure-triggered re-orchestration procedure removes an unavailable functional service and reconstructs the plan from the remaining catalogue, providing a mechanism for service-level recovery without changing the original request specification.
- A paired multi-seed evaluation protocol combines an exact complete-catalogue feasibility reference with controlled baselines, ablation analysis, single-service failure experiments, and request-volume timing. The feasibility reference distinguishes requests that are intrinsically infeasible in the generated catalogue from feasible requests missed because of bounded top-k search.

II. RELATED WORK

A. SMART GRID DATA MANAGEMENT AND DATA FABRIC

Smart grid data management has received increasing attention because modern grid applications increasingly depend on large-scale analytics and real-time sensor data for electricity generation, distribution, consumption, monitoring, protection, and control [16], [17]. Traditional approaches based on centralized databases, data warehouses, and manually maintained data catalogs can support basic data storage and retrieval. However, centralized or schema-bound datamanagement arrangements may be less adaptable to dynamically changing requirements, while interoperability problems persist across heterogeneous smart sensors and information systems from different grid and energy domains [17], [18]. Recent studies have therefore explored meta-data enrichment, ontology alignment, knowledge graphs, semantic descriptions, and natural-language or semantic query interfaces to improve data integration, consistent interpretation, organization, and discovery [12]. Data fabric has emerged as a modern data-management architecture that connects distributed data assets with metadata, governance mechanisms, data services, analytical capabilities, and automated access processes [18]. In a smart grid, such an architecture could serve as a unified data-supply layer for situational awareness, fault diagnosis, predictive maintenance, demand response, and operational analytics by making heterogeneous data consistently accessible to artificialintelligence and digital-twin applications [16], [19], [20].

However, the current data-fabric literature mainly emphasizes data integration, metadata-driven management, governance, accessibility, and workflow streamlining, whereas recent smartenergy semantic systems primarily emphasize ontology unification, data-to-ontology mapping, knowledge-graph construction, and semantic querying [12], [18], [21], [22]. The end-to-end transformation of grid business requirements into executable compositions of data services and computing services—subject to functional requirements and quality-of-service constraints—has not yet been sufficiently demonstrated in smart-grid-specific data-fabric research. The broader servicecomposition literature treats such executable workflow construction and constrained service selection as a separate optimization problem [15].

B. HETEROGENEOUS DATA ACCESS AND RETRIEVAL

Heterogeneous data access methods seek to expose autonomous, distributed, and heterogeneous sources through unified schemas and query interfaces. Recent surveys show that data federation systems can provide common access to relational databases, NoSQL stores, structured files, data warehouses, web services, and unstructured files, while data virtualization enables in-situ access through a common semantic layer without first moving all data into centralized storage [23]. Semantic data integration and semantic data management further employ mappings, metadata, ontologies, and knowledge graphs to improve integrated

querying, interoperability, and knowledge discovery across otherwise isolated sources [24], [25]. In smart-grid settings, big-data frameworks integrate operational data to support energy management and analytical applications [16], [26]. Knowledgegraph-based fault-handling systems can additionally transform unstructured dispatch procedures, fault-disposal plans, operating rules, and equipment information into structured knowledge networks that support fault-information analysis, diagnosis, and operational decision-making [27].

However, data-access and retrieval layers primarily answer queries or return retrieved records and supporting evidence; they do not necessarily produce a deployable business workflow. Data federation systems may provide source-level security functions such as authentication, authorization, auditing, encryption, and data masking, but their core task remains unified query answering rather than selecting and arranging the functional computing services required by a business task [23]. Service-composition research treats the selection and configuration of candidate services under functional requirements, quality-of-service attributes, cost, user preferences, and uncertainty as a separate problem [15]. Service-placement research further considers where service components should execute under resource-capacity, latency, availability, energy-consumption, and related operational constraints [9]. These distinctions motivate integrating heterogeneous data retrieval with business-semantic service composition, composition-level access-control validation, and resource-aware intelligent data provisioning rather than assuming that retrieval mechanisms alone can guarantee end-to-end executability and constraint satisfaction [9], [15], [23].

C. WORKFLOW SCHEDULING AND SERVICE ORCHESTRATION

Workflow scheduling and service composition have been extensively investigated in cloud, fog, service-computing, and distributed environments [28]. Recent systematic reviews show that workflow scheduling assigns interdependent tasks to heterogeneous computing resources and commonly optimizes makespan, latency, monetary cost, resource efficiency, reliability, scalability, and security [11]. Service-composition research addresses a related but distinct problem: selecting a configuration of functionally suitable candidate services while considering quality-of-service attributes, cost, user preferences, and environmental uncertainty [15]. The FAIR principles have also been applied to computational workflows to improve their findability, accessibility, interoperability, and reusability [29]. Importantly, many scheduling formulations take a predefined workflow or directed acyclic graph as their input and optimize task-to-resource mapping rather than deriving the workflow structure itself [11], [30]. In smart-grid applications, knowledge-graph research shows that heterogeneous operational records, equipment relationships, domain rules, and expert knowledge can support cross-business informa-

tion retrieval, fault diagnosis, fault disposal, and operational decision assistance [14]. This evidence suggests—but does not yet establish as a generally validated solution—that candidate data and computing services may need to be derived upstream from business semantics, data requirements, and operational constraints before conventional scheduling is applied [15], [30], [14].

Compared with predefined-workflow scheduling, business-oriented smart-grid data provisioning raises additional joint-selection requirements [31]. First, domain semantics and relationships among equipment, faults, operational records, and business rules are relevant to identifying the data and functions required by a task [14]. Second, data quality and timeliness must be evaluated because sensor failures, uncertain operating environments, missing or erroneous values, and outdated knowledge can degrade decision support and downstream analytics [14], [32]. Third, smart-grid data platforms combine heterogeneous data processing with analytical functions such as energy management, machine learning, anomaly or fault analysis, and operational decision support; therefore, data sources and computing services should be considered jointly rather than independently [16], [14]. Fourth, an executable orchestration plan must satisfy resource and quality-of-service feasibility as well as security, privacy, and deployment constraints. Recent workflow studies explicitly consider latency, cost, reliability, resource heterogeneity, deadlines, and restrictions on where sensitive tasks or data may execute [11], [30], [33]. Taken together, these findings motivate the development of a business-semantic-aware joint orchestration algorithm, although direct evidence for an end-to-end smart-grid method that simultaneously infers service chains and guarantees data-quality, resource, and permission feasibility remains limited [15], [16], [30]–[33].

D. MULTI-OBJECTIVE OPTIMIZATION FOR SERVICE PROVISIONING

Multi-objective optimization is widely used in quality-of-service-aware service composition, service placement, and resource scheduling. Common objectives include minimizing response latency, execution time, energy consumption, and monetary cost while maximizing reliability, availability, resource utilization, and overall quality of service [9], [15]. Weighted aggregation and Pareto-based search provide different mechanisms for balancing conflicting objectives, whereas evolutionary, heuristic, and metaheuristic algorithms are frequently used to explore large and computationally difficult service-selection and placement spaces [9], [34]. However, service provisioning in a smart-grid data-fabric environment cannot be reduced to numerical objective optimization alone. Power-domain knowledge-graph research indicates that operational records, equipment entities, fault relationships, business rules, and expert knowledge must be semantically organized to support interpretable retrieval, reasoning, and operational decision-making [14]. These findings suggest that smart-grid ser-

vice provisioning should combine multi-objective optimization with business-semantic matching and the joint representation of data services, computing services, and deployable resources [9], [15], [14]. BS-DCSO is therefore positioned as a bounded, feasibility-oriented orchestration method rather than as a global multi-objective optimizer. It combines exact functional matching, permission-aware candidate construction, QoS-based top-k pruning, business-context-aware completeplan scoring, exhaustive resource evaluation within the retained service space, and feasibility-first selection. The top-k restriction intentionally trades catalogue-level completeness for a bounded per-request search space, while the exact catalogue reference used in the experiments quantifies the resulting loss of feasible-instance coverage. The contribution is consequently evaluated in terms of end-to-end provisioning feasibility, constraint satisfaction, plan utility, failure recovery, and orchestration cost rather than global optimality. Comparisons with stronger published service-composition optimizers, global-optimality gaps on tractable instances, operator-facing interpretability, and high-concurrency distributed execution remain outside the present experimental scope [9], [34], [35].

III. PROBLEM FORMULATION

This section formulates the joint orchestration of data services, computing services, and resource nodes for smart-grid operational analytics. Given a business request, the objective is to construct a complete provisioning plan that covers all required functional types while satisfying data-quality, latency, reliability, cost, permission, and resource-capacity constraints. Among candidate plans with different degrees of feasibility, the formulation further defines a feasibility-first preference rule for selecting the final plan.

A. REQUEST, SERVICE, AND PLAN MODELS

Let $\mathcal{Q}$ denote the set of business requests. Each request $q \in \mathcal{Q}$ is represented as

$$q = \{s, o, r, D_q, C_q, L_q, Q_q, R_q, K_q, P_q\}. \quad (1)$$

where s denotes the business scenario, o the grid object, and r the geographical or operational region. D_q is the set of required data types and C_q is the set of required computing-service types. The operational constraints are represented by the maximum admissible latency L_q , minimum data-quality threshold Q_q , minimum reliability threshold R_q , maximum cost K_q , and requester permission level P_q .

Let $\mathcal{D}$ denote the data-service catalogue. Each data service $d \in \mathcal{D}$ is described by

$$d = \{ \text{type}(d), \text{region}(d), \text{quality}(d), \text{reliability}(d), \text{latency}(d), \text{cost}(d), \text{permission}(d) \}. \quad (2)$$

Here, $\text{type}(d)$ identifies the functional data type supplied by the service, while the remaining attributes describe its regional association, data quality, reliability, access latency, cost, and required permission level.

Let $\mathcal{C}$ denote the computing-service catalogue. Each computing service $c \in \mathcal{C}$ is represented by

$$c = \{ \text{type}(c), \text{scenarios}(c), \text{accuracy}(c), \text{reliability}(c), \text{latency}(c), \text{cost}(c), \text{cpu}(c), \text{mem}(c), \text{gpu}(c), \text{permission}(c) \}. \quad (3)$$

The set $\text{scenarios}(c)$ specifies the business scenarios supported by service c , while $\text{cpu}(c)$, $\text{mem}(c)$, and $\text{gpu}(c)$ represent its resource requirements.

Let $\mathcal{R}$ denote the set of available resource nodes. Each node $u \in \mathcal{R}$ is characterized by

$$u = \{ \text{capacity}_{\text{cpu}}(u), \text{capacity}_{\text{mem}}(u), \text{capacity}_{\text{gpu}}(u), \text{reliability}(u), \ell(u), \text{region}(u), \phi(u) \}. \quad (4)$$

where $\ell(u)$ is the current load ratio and $\phi(u)$ is the node-specific cost factor.

For request q , a complete orchestration plan is represented as

$$P = \{D_P, C_P, u_P\}, \quad (5)$$

where $D_P \subseteq \mathcal{D}$ is the selected set of data services, $C_P \subseteq \mathcal{C}$ is the selected set of computing services, and $u_P \in \mathcal{R}$ is the resource node on which the selected computing services are provisioned. For notational simplicity, u is used below to denote the selected node u_P when the plan P is clear from context.

A complete plan must provide all data and computing functions required by the request:

$$D_q \subseteq \{ \text{type}(d) \mid d \in D_P \}, \quad (6)$$

$$C_q \subseteq \{ \text{type}(c) \mid c \in C_P \}. \quad (7)$$

The implemented orchestration procedure selects one service for each required functional type. Therefore, when complete functional coverage is achieved, $|D_P| = |D_q|$ and $|C_P| = |C_q|$.

B. PLAN-LEVEL QUALITY, LATENCY, RELIABILITY, AND COST

Let $m_D = |D_q|$, $m_C = |C_q|$, and define the clipping operator $[z]_0^1 = \min\{1, \max\{0, z\}\}$. The mean data quality of a complete plan is defined as

$$DQ(P) = \frac{1}{m_D} \sum_{d \in D_P} \text{quality}(d), \quad (8)$$

and the mean computing accuracy is

$$CA(P) = \frac{1}{m_C} \sum_{c \in C_P} \text{accuracy}(c). \quad (9)$$

A soft quality score combines these two attributes:

$$Q(P) = 0.65DQ(P) + 0.35CA(P). \quad (10)$$

The distinction between $DQ(P)$ and $Q(P)$ is important. $Q(P)$ contributes to soft complete-plan utility, whereas the

hard data-quality constraint is imposed only on $DQ(P)$. Computing accuracy therefore influences plan preference but is not itself used as a hard feasibility threshold.

For a plan deployed on resource node u with load ratio $\ell(u)$, the implemented latency estimate is

$$\text{latency}(P) = \sum_{d \in D_P} \text{latency}(d) + (1 + \ell(u)) \sum_{c \in C_P} \text{latency}(c). \quad (11)$$

The first term represents the cumulative data-service latency. The second term represents computing-service latency adjusted by the current resource-node load. This quantity is used as the orchestration-level latency estimate throughout the evaluation.

The plan-level reliability score is defined as the geometric mean of the reliability values of all selected functional services and the selected resource node. For notational simplicity, let $R(x)$ denote the reliability of service or resource node x . Accordingly, the plan-level reliability is given by

$$\text{Rel}(P) = \left(\prod_{d \in D_P} R(d) \prod_{c \in C_P} R(c) R(u) \right)^{\frac{1}{m_D + m_C + 1}}. \quad (12)$$

This aggregation is used consistently for both complete-plan scoring and reliability-feasibility evaluation.

The estimated plan cost is

$$\text{cost}(P) = \phi(u) \left(\sum_{d \in D_P} \text{cost}(d) + \sum_{c \in C_P} \text{cost}(c) \right), \quad (13)$$

where $\phi(u)$ adjusts the aggregate functional-service cost according to the selected resource node.

To express latency and cost as utility components with larger values indicating better satisfaction, they are normalized as

$$T(P) = \left[1 - \frac{\text{latency}(P)}{L_q} \right]_0^1, \quad (14)$$

$$\text{CostSat}(P) = \left[1 - \frac{\text{cost}(P)}{K_q} \right]_0^1. \quad (15)$$

Both terms are bounded within $[0, 1]$. They are used only as soft utility components; the original latency and cost quantities are separately checked against their hard thresholds.

C. RESOURCE FITNESS AND CAPACITY FEASIBILITY

For each resource dimension $z \in \{\text{cpu}, \text{mem}, \text{gpu}\}$, the currently available capacity of node u is defined as

$$\text{avail}_z(u) = \text{capacity}_z(u)(1 - \ell(u)). \quad (16)$$

The aggregate resource demand of the selected computing services is

$$\text{req}_z(C_P) = \sum_{c \in C_P} \text{req}_z(c). \quad (17)$$

The corresponding soft resource-fit component is

$$\text{Fit}_z(C_P, u) = \min \left\{ 1, \frac{\text{avail}_z(u)}{\max\{\text{req}_z(C_P), 10^{-9}\}} \right\}. \quad (18)$$

When none of the selected computing services requires a GPU, $\text{Fit}_{\text{gpu}}(C_P, u)$ is explicitly set to one. The aggregate resource-fitness score is

$$F(P) = 0.45 \text{Fit}_{\text{cpu}}(C_P, u) + 0.35 \text{Fit}_{\text{mem}}(C_P, u) + 0.20 \text{Fit}_{\text{gpu}}(C_P, u). \quad (19)$$

Resource fitness measures how well the selected node accommodates the computing-service demand and contributes to soft plan utility. The CPU, memory, and GPU coefficients are fixed soft-preference weights rather than learned or optimized parameters. They do not replace resource-capacity feasibility. A plan satisfies the hard capacity constraint only if

$$\text{req}_z(C_P) \leq \text{avail}_z(u), \quad z \in \{\text{cpu}, \text{mem}, \text{gpu}\}. \quad (20)$$

Thus, a resource node with a favorable fitness score may still be rejected if any required resource dimension exceeds its available capacity.

D. CONTEXTUAL RELEVANCE AND COMPLETE-PLAN UTILITY

BS-DCSO uses lightweight textual business-context similarity to differentiate functionally compatible service compositions. Let $\text{Sem}(q, x)$ denote the TF-IDF cosine similarity between request q and functional service x . The contextual relevance of a complete plan is defined as the mean similarity over all selected data and computing services:

$$S(P) = \frac{\sum_{d \in D_P} \text{Sem}(q, d) + \sum_{c \in C_P} \text{Sem}(q, c)}{m_D + m_C}. \quad (21)$$

The quantity $S(P)$ represents textual business-context consistency rather than ontology-level semantic equivalence. Functional compatibility is determined separately through exact service-type matching.

The complete-plan utility implemented by BS-DCSO is

$$U(P) = 0.28S(P) + 0.18Q(P) + 0.16T(P) + 0.16 \text{Rel}(P) + 0.12F(P) + 0.10 \text{CostSat}(P). \quad (22)$$

The six weights sum to one and remain fixed throughout all experiments. They are implementation-level soft-preference weights rather than learned parameters and are not presented as universally optimal. Utility is used to distinguish candidate plans after considering feasibility; consequently, these weights do not relax any hard constraint and cannot permit a high-scoring infeasible plan to dominate a feasible alternative.

E. CANDIDATE SPACE AND FEASIBILITY

To bound the service-composition search, BS-DCSO retains at most $k_d = 2$ candidates for each required data type and at most $k_c = 3$ candidates for each required computing-service type. These values are fixed search-budget settings used throughout the study to keep complete-plan enumeration bounded; they are not claimed to be optimal for all catalogues or workloads.

Let K_t^D denote the retained data-service list for required data type $t \in D_q$, and let K_g^C denote the retained computing-service list for required computing type $g \in C_q$. The candidate-plan space can then be expressed as

$$\mathcal{P}_A(q) = \{(D_P, C_P, u) \mid D_P \in \prod_{t \in D_q} K_t^D, C_P \in \prod_{g \in C_q} K_g^C, u \in \mathcal{R}\}. \quad (23)$$

For a request containing m_D required data types, m_C required computing types, and $N_R = |\mathcal{R}|$ resource nodes, the size of the implemented candidate-plan space therefore satisfies

$$|\mathcal{P}_A(q)| \leq 2^{m_D} 3^{m_C} N_R. \quad (24)$$

The search is bounded solely through the per-type candidate limits k_d and k_c . No additional service-combination budget, resource top-k restriction, or independent resource-fitness threshold is applied. Consequently, every retained functional-service combination is evaluated on every available resource node.

A candidate plan is feasible only if it provides complete functional coverage and satisfies all hard operational constraints. The feasible set is therefore

$$\begin{aligned} \mathcal{F}_A(q) = \{P \mid P \in \mathcal{P}_A(q), \\ D_q \subseteq T(D_P), C_q \subseteq T(C_P), \\ DQ(P) \geq Q_q, \text{ latency}(P) \leq L_q, \\ \text{Rel}(P) \geq R_q, \text{ cost}(P) \leq K_q, \\ \text{permission}(x) \leq P_q, \forall x \in D_P \cup C_P, \\ \text{req}_z(C_P) \leq \text{avail}_z(u_P), \\ \forall z \in \{\text{cpu}, \text{mem}, \text{gpu}\}\}. \end{aligned} \quad (25)$$

Permission levels are ordered from less restrictive to more restrictive. Hence, $\text{permission}(x) \leq P_q$ indicates that the requester has sufficient clearance to access functional service x . Permission is treated as a hard constraint and is not compensated for by other utility components.

The feasible set $\mathcal{F}_A(q)$ is defined with respect to the retained top-k candidate space rather than the complete service catalogue. Therefore, $\mathcal{F}_A(q) = \emptyset$ does not necessarily imply that the original request is catalogue-infeasible; a feasible composition may have been excluded during top-k pruning. This distinction motivates the complete-catalogue feasibility reference used in the experimental evaluation.

F. FEASIBILITY-FIRST PLAN SELECTION

BS-DCSO uses a lexicographic preference rule so that hard-constraint feasibility takes precedence over soft utility. Let $V_{\text{all}}(P)$ denote the number of violated constraint categories among data-type coverage, computing-type coverage, data quality, latency, reliability, cost, permission, and resource capacity. A resource-capacity violation is recorded when at least one of the CPU, memory, or GPU capacity constraints is not satisfied.

Let $V_{\text{crit}}(P)$ count the subset of violations associated with data-type coverage, computing-type coverage, permission, and resource capacity. These constraints are treated as critical because they determine whether the required functions can be accessed and physically provisioned.

The selected plan is

$$P^* = \arg \max_{P \in \mathcal{P}_A(q)} (\mathbb{I}[V_{\text{all}}(P) = 0], -V_{\text{crit}}(P), -V_{\text{all}}(P), U(P)). \quad (26)$$

where the tuple is compared lexicographically from left to right. The first component distinguishes feasible from infeasible plans. The second and third components minimize critical and total violations, respectively, and the final component maximizes complete-plan utility among otherwise equivalent candidates.

Consequently, every feasible candidate plan dominates every infeasible candidate regardless of utility. If multiple feasible plans exist, the plan with the highest utility is selected. If no feasible plan exists in the retained candidate space, BS-DCSO first selects a fallback with the fewest critical violations, then the fewest total violations, and finally the highest utility. If a required functional type has no available catalogue service and the candidate-plan space cannot be constructed, the implementation returns an empty fallback plan so that the missing functional coverage is represented explicitly by the violation indicators.

IV. PROPOSED BS-DCSO ALGORITHM

BS-DCSO transforms a structured smart-grid business request into a complete provisioning plan consisting of data services, computing services, and a deployment resource node. As illustrated in Figure 1, the procedure contains five main stages: contextual similarity calculation, exact-type and permission-aware candidate construction, QoS-oriented top-k pruning, joint service-combination and resource-node evaluation, and feasibility-first plan selection. Contextual TF-IDF similarity is used only during complete-plan evaluation and does not determine the top-k candidate space. When a selected functional service becomes unavailable, the failed service is excluded from the catalogue and the same orchestration procedure is executed again to search for a replacement plan.

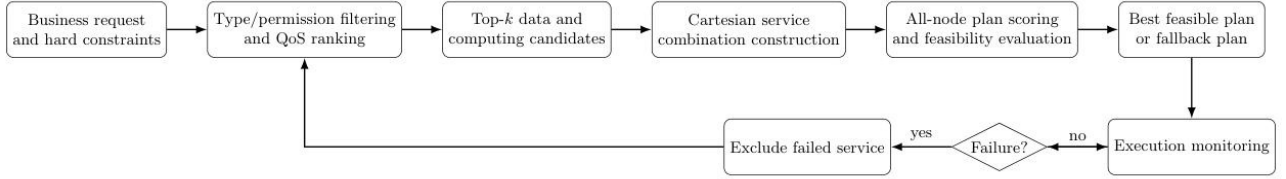


FIGURE 1. Workflow of BS-DCSO. Exact functional matching, permission availability, and QoS attributes determine the retained top-k candidate space. Contextual TF-IDF similarity contributes only to complete-plan scoring. Every retained service composition is evaluated on every resource node, and functional-service failure triggers re-orchestration on the reduced catalogue.

A. BUSINESS-CONTEXT REPRESENTATION AND SIMILARITY

Each business request is represented by both structured attributes and a generated textual description. The structured attributes include the business scenario, grid object, region, required data and computing types, time window, maximum latency, minimum data quality, minimum reliability, maximum cost, and permission level. The textual representation provides a lightweight mechanism for distinguishing services associated with different smart-grid business contexts while the structured fields determine functional and operational feasibility.

The TF-IDF vectorizer is fitted once on the fixed data- and computing-service catalogue. Subsequent requests are treated as unseen queries and do not modify the fitted vocabulary or inverse-document-frequency values. Lowercase word unigrams and bigrams are used. Service text concatenates, where available, the service name, functional type, region, tags, required data types, and supported scenarios.

For request q and service x , contextual similarity is calculated as

$$\text{Sem}(q, x) = \cos(v_q, v_x), \quad (27)$$

where v_q and v_x are their TF-IDF vectors. This measure captures lexical business-context consistency rather than ontology-level semantic equivalence or logical reasoning.

For each request, similarity values are calculated for all catalogue services and stored in request-specific similarity maps. Importantly, these values are deliberately excluded from top-k candidate pruning. The full BS-DCSO model and the semantic-ablation variant therefore operate on the same retained candidate space. Contextual similarity affects the final decision only through the complete-plan relevance score $S(P)$ and utility $U(P)$ defined in Section 3.

B. PERMISSION-AWARE DATA-SERVICE CANDIDATE SELECTION

For each required data type $t \in D_q$, BS-DCSO first constructs the exact-type candidate set $A_t^D = \{d \in \mathcal{D} : \text{type}(d) = t\}$. The use of exact-type matching ensures that contextual similarity cannot substitute for functional compatibility. Permission filtering is then applied. Let $G_t^D = \{d \in A_t^D : \text{permission}(d) \leq P_q\}$. The ranking pool is defined as

$$H_t^D = \begin{cases} G_t^D, & G_t^D \neq \emptyset, \\ A_t^D, & G_t^D = \emptyset. \end{cases} \quad (28)$$

Thus, permission-feasible candidates are used whenever at least one exists. If no exact-type service satisfies the requester permission level, the complete exact-type set is retained so that the algorithm can return an explicit permission-violating fallback rather than terminating without a diagnostic plan. If $A_t^D = \emptyset$, no service can provide the required data type and an empty fallback plan is returned.

For any functional service x , latency and cost fitness are defined as

$$\text{LatFit}(q, x) = \left[1 - \frac{\text{latency}(x)}{L_q} \right]_0^1, \quad (29)$$

$$\text{CostFit}(q, x) = \left[1 - \frac{\text{cost}(x)}{K_q} \right]_0^1. \quad (30)$$

Regional compatibility for data service d is

$$\text{RegionFit}(q, d) = \begin{cases} 1, & \text{region}(d) = \text{region}(q), \\ 0.75, & \text{otherwise.} \end{cases} \quad (31)$$

The implemented data-service ranking score is

$$\begin{aligned} \text{Rank}_d(q, d) = & 0.20 \text{quality}(d) + 0.16 \text{reliability}(d) \\ & + 0.14 \text{LatFit}(q, d) + 0.10 \text{RegionFit}(q, d) \\ & + 0.10 \text{CostFit}(q, d). \end{aligned} \quad (32)$$

The ranking score is used only to order services within an exact-type candidate pool and is not interpreted as a normalized complete-plan utility. Contextual similarity is intentionally excluded from this stage. Consequently, the coefficients in $\text{Rank}_d(q, d)$ are relative ranking weights and are not required to sum to one. The highest-ranked $k_d = 2$ services in H_t^D are retained as K_t^D .

C. PERMISSION-AWARE COMPUTING-SERVICE CANDIDATE SELECTION

For each required computing-service type $g \in C_q$, the exact-type candidate set is $A_g^C = \{c \in \mathcal{C} : \text{type}(c) = g\}$. Permission handling follows the same rule as for data services. Let $G_g^C = \{c \in A_g^C : \text{permission}(c) \leq P_q\}$, and define

$$H_g^C = \begin{cases} G_g^C, & G_g^C \neq \emptyset, \\ A_g^C, & G_g^C = \emptyset. \end{cases} \quad (33)$$

If $A_g^C = \emptyset$, the required computing function cannot be provided and an empty fallback plan is returned.

Because computing services may be designed for different operational applications, scenario compatibility is defined as

$$\text{ScenarioFit}(q, c) = \begin{cases} 1, & \text{scenario}(q) \in \text{supportedScenarios}(c), \\ 0.6, & \text{otherwise.} \end{cases} \quad (34)$$

Computing-service candidates are ranked according to

$$\begin{aligned} \text{Rank}_c(q, c) = & 0.14 \text{ScenarioFit}(q, c) \\ & + 0.20 \text{accuracy}(c) + 0.14 \text{reliability}(c) \\ & + 0.12 \text{LatFit}(q, c) \\ & + 0.12 \text{CostFit}(q, c). \end{aligned} \quad (35)$$

As with data-service ranking, $\text{Rank}_c(q, c)$ is a relative candidate-ordering criterion rather than a normalized plan utility. Contextual similarity does not participate in pruning and is introduced only after complete service combinations have been constructed. The highest-ranked $k_c = 3$ candidates in H_g^C are retained as K_g^C .

D. JOINT SERVICE COMPOSITION AND RESOURCE EVALUATION

After candidate pruning, BS-DCSO constructs complete functional-service combinations using the Cartesian products $D_P \in \prod_{t \in D_q} K_t^D$, $C_P \in \prod_{g \in C_q} K_g^C$. Each pair (D_P, C_P) contains one retained service for every required data and computing type. Unlike the functional-service catalogues, the resource-node set is not pruned. Every retained service composition is therefore evaluated on every node $u \in \mathcal{R}$.

For each service composition, its contextual relevance is calculated from the precomputed similarity maps:

$$S = \frac{\sum_{d \in D_P} \text{Sem}(q, d) + \sum_{c \in C_P} \text{Sem}(q, c)}{|D_P| + |C_P|}. \quad (36)$$

For each resource node, the procedure then evaluates the complete plan $P = \{D_P, C_P, u\}$ using the quality, latency, reliability, cost, resource-fitness, permission, capacity, and utility definitions in Section 3. Resource evaluation is exhaustive within the retained functional-service space: no resource ranking, resource top-k truncation, or independent resource-fitness threshold is applied. This design separates two forms of search reduction. The potentially large functional-service catalogue is bounded through the per-type limits $k_d = 2$ and $k_c = 3$, whereas resource placement remains exhaustive for every retained service composition. Consequently, a candidate composition is not rejected simply because it performs poorly on an initially selected node; it can still be evaluated on alternative nodes with different capacities, loads, reliability values, and cost factors.

E. FEASIBILITY-FIRST PLAN SELECTION AND FALLBACK HANDLING

Every evaluated plan is compared using the feasibility-first lexicographic rule defined in Section 3. For convenience, let

$$\text{Key}(P) = (\mathbb{I}[V_{\text{all}}(P) = 0], -V_{\text{crit}}(P), -V_{\text{all}}(P), U(P)). \quad (37)$$

For two candidate plans P_1 and P_2 , BS-DCSO prefers P_1 whenever $\text{Key}(P_1) >_{\text{lex}} \text{Key}(P_2)$. The first component guarantees that any fully feasible plan is preferred to every infeasible plan. When several feasible plans exist, their first three key components are identical and the plan with the highest complete-plan utility is selected.

If the retained candidate space contains no feasible plan, the same key provides deterministic fallback handling. BS-DCSO first minimizes critical violations involving functional coverage, permission, and resource capacity, then minimizes the total number of violated constraint categories, and finally uses utility to distinguish otherwise equivalent fallback plans. This fallback mechanism preserves information about why a request cannot be fully provisioned instead of reducing all unsuccessful requests to an undifferentiated failure state.

F. FEEDBACK-DRIVEN RE-ORCHESTRATION

The initial plan may become invalid if a selected data or computing service becomes unavailable during execution. BS-DCSO addresses this event through service-exclusion-based re-orchestration. Let x_f denote the identifier of the failed functional service. The reduced catalogues are $\mathcal{D}' = \{d \in \mathcal{D} : \text{id}(d) \neq x_f\}$, $\mathcal{C}' = \{c \in \mathcal{C} : \text{id}(c) \neq x_f\}$. The original business request and resource-node set remain unchanged, and BS-DCSO is executed again using $\mathcal{D}'$, $\mathcal{C}'$, and $\mathcal{R}$. Because candidate construction, service composition, and resource evaluation are repeated, the replacement plan may use alternative functional services, a different resource node, or both.

Re-orchestration succeeds when the reduced catalogue contains a candidate that satisfies all original hard constraints. If no such candidate exists, the feasibility-first rule returns the best available fallback according to the same violation hierarchy used during initial orchestration. The mechanism therefore handles the loss of an individual functional-service instance without requiring a separately defined recovery objective. The experimental protocol used to generate and evaluate service failures is described separately in Section 5.

V. EXPERIMENTAL EVALUATION

This section evaluates BS-DCSO in terms of complete-plan feasibility, constraint satisfaction, plan utility, failure recovery, component contribution, and orchestration runtime. Because no standardized benchmark currently exists for the joint data-service, computing-service, and resource-node orchestration problem considered in this study, a controlled simulation benchmark is constructed to permit repeatable

comparisons under identical request, service, resource, and failure conditions [36], [37].

A. BENCHMARK DESIGN AND EXPERIMENTAL PROTOCOL

The benchmark contains four interacting entity classes: business requests, data services, computing services, and resource nodes. Five smart-grid operational scenarios are considered: situational awareness, load forecasting, renewable-energy monitoring, fault analysis, and equipment diagnosis. Each standard seed-specific benchmark instance contains 300 business requests, 300 data services, 30 computing services, and 18 resource nodes. These catalogue sizes are experimental settings rather than standardized smart-grid benchmark dimensions.

Table I summarizes the principal algorithmic and experimental settings.

The synthetic service-catalogue and resource-node generation rules are summarized in Table II. Normal draws denoted by $\mathcal{N}(\mu, \sigma)$ use the stated standard deviation. The operator $\text{clip}_{[0,1]}$ truncates values to the interval $[0, 1]$, whereas $\max\{a, X\}$ imposes only a lower bound.

Table III reports the business-request generation rules.

The synthetic construction is intended to provide controlled variation in service quality, resource capacity, permissions, latency, reliability, and cost across multiple smart-grid operational contexts. It does not imply that the resulting marginal distributions or catalogue dimensions reproduce those of a specific deployed utility platform.

B. COMPLETE-CATALOGUE FEASIBILITY REFERENCE

Overall provisioning failure may arise for two fundamentally different reasons: either no hard-constraint-feasible plan exists in the generated catalogue, or a feasible plan exists but is missed by the bounded orchestration procedure. To distinguish these cases, an exact complete-catalogue feasibility reference is evaluated for every request.

The reference selects one service for each required data and computing type and one resource node from the complete generated catalogues. Permission-infeasible services are removed, Pareto pruning is applied on hard-feasibility dimensions, and depth-first search is performed using quality, latency, cost, and reliability bounds. Unlike BS-DCSO, the reference does not apply the top-k candidate restriction, does not use TF-IDF contextual relevance, and does not optimize complete-plan utility. Its sole purpose is to determine whether at least one plan satisfying all hard constraints exists in the finite generated catalogue.

The reference should therefore be interpreted as a feasibility certificate for the generated benchmark rather than as an optimal orchestration algorithm. It does not establish that the finite synthetic catalogue is complete or representative of a deployed smart-grid data fabric.

C. RANDOM SEEDS, STATISTICAL ANALYSIS, AND IMPLEMENTATION ENVIRONMENT

The main comparison, ablation experiment, and failure-recovery experiment use the same ten random seeds $\{41, 42, 43, 44, 45, 46, 47, 48, 49, 50\}$.

For a given seed, all evaluated methods and variants receive identical service catalogues, resource nodes, business requests, complete-catalogue feasibility labels, and failure specifications. This common-random-number design enables differences to be calculated within the same seed-specific benchmark instance and reduces variation caused by independently generated workloads.

For a metric with n valid seeds, results are reported as the sample mean and two-sided 95% Student's t confidence interval:

$$\bar{x} \pm t_{0.975, n-1} \frac{s}{\sqrt{n}}, \quad (38)$$

where s denotes the sample standard deviation. For the ten-seed experiments, $t_{0.975, 9} = 2.262$.

Metrics with fewer than three valid seeds are reported as N/A. For method comparisons and ablation analyses, differences are first calculated within each common seed and then aggregated using the same confidence-interval procedure.

D. IMPLEMENTATION ENVIRONMENT.

All experiments were executed on a single server running Ubuntu 22.04.5 LTS with two Intel Xeon Silver 4310 CPUs at 2.10 GHz, providing 24 physical CPU cores and 48 logical threads, and 251 GiB of system memory. The server also contained four NVIDIA GeForce RTX 4090 GPUs with approximately 24 GB of device memory per GPU. The orchestration implementation does not use GPU acceleration, and these GPUs were not involved in the reported timing measurements.

The experimental software environment used Python 3.10.20, NumPy 2.2.6, pandas 2.3.3, SciPy 1.15.3, scikit-learn 1.7.2, NetworkX 3.4.2, PyYAML 6.0.3, and Matplotlib 3.10.9. All compared methods were executed in the same software and hardware environment. Reported runtime measures orchestration computation only and excludes benchmark generation, exact-catalogue feasibility evaluation, metric aggregation, service deployment, network data transfer, analytical-task execution, and state migration. TF-IDF fitting is also excluded where specified by the corresponding timing protocol.

E. COMPARED METHODS

The primary multi-seed experiment compares BS-DCSO with five controlled baseline implementations:

- GreedySemanticMatching applies semantic-first local selection, preferring services with strong request-service TF-IDF similarity without complete-plan joint optimization.
- QoSWeightedGreedy applies QoS-oriented greedy selection using service quality, reliability, latency, cost, con-

TABLE I. BS-DCSO configuration and experimental protocol.

Setting	Configuration and role
Data-service pruning	Retain at most $k_d = 2$ QoS-ranked candidates for each required data type.
Computing-service pruning	Retain at most $k_c = 3$ QoS-ranked candidates for each required computing type.
Plan enumeration	Enumerate the Cartesian product of all retained per-type service lists and evaluate every one of the 18 resource nodes. No additional service-combination budget, resource top-k, or resource-fitness threshold is used.
Complete-plan utility	Weights (0.28, 0.18, 0.16, 0.16, 0.12, 0.10) for contextual relevance, quality, latency satisfaction, reliability, resource fitness, and cost satisfaction, respectively.
Resource fitness	Weights (0.45, 0.35, 0.20) for CPU, memory, and GPU fitness. Hard capacity feasibility is checked separately in all three resource dimensions.
Contextual model	Lowercase TF-IDF word unigrams and bigrams fitted once on the fixed service catalogue. Requests are treated as unseen queries. TF-IDF similarity contributes to complete-plan utility but not to top-k candidate pruning.
Feasibility reference	Exact complete-catalogue search over exact-type, permission-feasible services and all resource nodes. BS-DCSO top-k pruning is not applied and utility is not optimized.
Multi-seed protocol	Seeds 41–50 for the main comparison, ablation analysis, and failure-recovery experiment. Within each seed, all evaluated methods use the same requests, service catalogues, resource nodes, oracle labels, and request-level failure specifications.
Failure injection	One failed functional requirement is generated for every request. The requirement is sampled uniformly from the concatenation of its required data and computing types and is shared across evaluated methods. Recovery is evaluated only when the initial plan is feasible and the failed functional requirement is implemented by that plan.
Request-volume timing	Request volumes of 100, 300, 500, and 1000 using benchmark seed 42, with one warm-up execution followed by five timed repetitions. Timing covers orchestration computation only.

TABLE II. Synthetic service-catalogue and resource-node generation rules.

Entity	Size	Generation rule
Data services	300	Five regions $\times$ ten data types $\times$ three variants $\times$ two replicas. Base quality is $\text{clip}_{[0,1]} \mathcal{N}(0.84, 0.08)$ with replica noise $\mathcal{N}(0, 0.03)$; base reliability is $\text{clip}_{[0,1]} \mathcal{N}(0.88, 0.07)$ with replica noise $\mathcal{N}(0, 0.04)$. Latency is $\max\{50, \mathcal{N}(350, 120)\}$ ms, with an additional 120 ms for alarm logs and maintenance work orders. Cost is $\max\{1, \mathcal{N}(8, 3)\}$. Permission levels 1, 2, and 3 have probabilities 0.50, 0.333, and 0.167, respectively.
Computing services	30	Ten computing types $\times$ three variants. Latency is $\max\{100, \mathcal{N}(900, 300)\}$ ms; accuracy is $\text{clip}_{[0,1]} \mathcal{N}(0.86, 0.06)$; reliability is $\text{clip}_{[0,1]} \mathcal{N}(0.88, 0.07)$; and cost is $\max\{2, \mathcal{N}(15, 5)\}$. CPU demand is $\max\{0.5, \mathcal{N}(2.5, 0.8)\}$, memory demand is $\max\{1, \mathcal{N}(6, 2)\}$ GB, and GPU demand is $\max\{0, G + \mathcal{N}(0, 0.15)\}$, where G is sampled from $\{0, 0, 0, 1\}$. Permission levels 1, 2, and 3 have probabilities 0.40, 0.40, and 0.20, respectively.
Resource nodes	18	Five regional locations and one cloud location are used, with three resource nodes generated for each location. For a regional node, CPU, memory, and GPU capacities are sampled from $\{4, 8, 16\}$, $\{8, 16, 32\}$ GB, and $\{0, 0, 1\}$, respectively. For a cloud node, the corresponding capacities are sampled from $\{8, 16, 32, 64\}$, $\{32, 64, 128, 256\}$ GB, and $\{0, 1, 2, 4\}$.

TABLE III. Synthetic business-request generation rules.

Request field	Generation rule
Business context	Scenario, region, and grid object are sampled uniformly from five scenarios, five regions, and eight object types.
Functional requirements	Scenario-specific data and computing requirements are fixed by the generator. With probability 0.25, one uniformly sampled additional data type is appended if it is not already required.
Maximum latency	$\max\{2500, \mathcal{N}(6200, 900)\}$ ms.
Minimum data quality	$\text{clip}_{[0,1]} \mathcal{N}(0.78, 0.08)$.
Minimum reliability	$\text{clip}_{[0,1]} \mathcal{N}(0.78, 0.07)$.
Maximum cost	$\max\{20, \mathcal{N}(70, 20)\}$.
Permission level	Levels 1, 2, and 3 with probabilities 0.40, 0.40, and 0.20, respectively.
Time window	Uniformly sampled from $\{1, 6, 12, 24, 48, 168\}$ hours.

textual compatibility, permission, and resource-related attributes without enumerating the BS-DCSO top-k Cartesian plan space.

- **ResourceFirstScheduling** first prioritizes a resource node and then incrementally constructs the required functional-service composition subject to resource-capacity constraints.
- **ShortestLatency** applies latency-oriented greedy selection without complete-plan balancing of quality, reliability, cost, contextual relevance, permission, and resource feasibility.
- **FeasibleShortestLatency** incorporates feasibility checks before latency-oriented plan selection and therefore provides a stronger feasibility-aware latency reference than **ShortestLatency**.

These baselines represent semantic-first, QoS-greedy, resource-first, latency-first, and feasibility-aware latency decision paradigms. They are controlled implementations rather than exact reproductions of specific published algorithms. All methods operate under the same requests, catalogues, resource nodes, hard constraints, feasibility labels, and random seeds. Consequently, the comparison evaluates the effect of different orchestration strategies within the controlled benchmark and should not be interpreted as establishing superiority over the broader published service-composition literature.

F. EVALUATION METRICS

The evaluation uses the following metrics:

Provisioning Success Rate (PSR) is the proportion of

all requests for which the evaluated method generates a complete plan satisfying every hard constraint:

$$\text{PSR} = \frac{N_{\text{successful}}}{N_{\text{all}}}. \quad (39)$$

Oracle Feasibility Rate (OFR) is the proportion of all requests for which the complete-catalogue reference confirms the existence of at least one hard-constraint-feasible plan:

$$\text{OFR} = \frac{N_{\text{catalogue-feasible}}}{N_{\text{all}}}. \quad (40)$$

Feasible-Instance Success Rate (FISR) is the proportion of oracle-feasible requests for which the evaluated method finds a feasible plan:

$$\text{FISR} = \frac{\sum_i \mathbb{I}[\text{success}_i = 1 \wedge \text{oracle}_i = 1]}{\sum_i \mathbb{I}[\text{oracle}_i = 1]}. \quad (41)$$

FISR separates failures of the bounded orchestration procedure from requests that are infeasible in the complete finite catalogue.

Constraint Violation Rate is the number of violated applicable hard-constraint categories divided by the number of applicable hard-constraint categories for each request, averaged over all requests.

End-to-End Utility is the complete-plan utility $U(P)$ averaged over all returned plans, including infeasible fallback plans.

Feasible-Plan Utility (FPU) is the complete-plan utility averaged only over successfully provisioned requests:

$$\text{FPU} = \frac{1}{N_{\text{successful}}} \sum_{i: P_i \text{ feasible}} U(P_i). \quad (42)$$

Semantic Relevance is the mean TF-IDF contextual similarity of the selected functional services. It measures lexical business-context consistency and is not a feasibility criterion.

Service Selection Quality is a diagnostic QoS-oriented score calculated by the evaluation implementation from normalized service-level quality attributes used by the corresponding selection procedure. It is reported as a local-selection diagnostic and does not determine plan feasibility.

Average Runtime per Request is the wall-clock orchestration time divided by the number of processed requests. It excludes deployment, data transfer, analytical-model execution, and state migration.

Recovery Success Rate is the proportion of eligible single-service failure trials for which a hard-constraint-feasible replacement plan is generated after re-orchestration.

G. MULTI-SEED COMPARISON WITH CONTROLLED BASELINES

The oracle feasibility rate is a property of the generated benchmark rather than of an individual orchestration method. Across seeds 41–50, the common OFR is 0.5867 ± 0.1214 .

Tables IV and V report the multi-seed comparison. Unless otherwise noted, values are the mean $\pm$ two-sided 95%

Student's t confidence interval over the ten common random seeds.

BS-DCSO achieves the highest provisioning success rate, 0.4897 ± 0.1139 , compared with 0.3473 ± 0.0715 for ResourceFirstScheduling, 0.2757 ± 0.1055 for QoSWeightedGreedy, and 0.2710 ± 0.0662 for FeasibleShortestLatency. Its FISR is also the highest at 0.8320 ± 0.0562 , indicating that the bounded top-k joint search recovers a large proportion of requests for which the complete-catalogue reference confirms feasibility.

BS-DCSO also yields the lowest aggregate constraint violation rate, 0.0809 ± 0.0213 , followed by ResourceFirstScheduling at 0.1073 ± 0.0161 . These results indicate that its principal advantage lies in constructing complete plans that jointly satisfy heterogeneous operational constraints rather than maximizing an individual local ranking criterion.

The local diagnostic metrics expose this distinction. GreedySemanticMatching achieves the highest TF-IDF semantic relevance, 0.2871 ± 0.0018 , whereas BS-DCSO achieves 0.2697 ± 0.0019 . QoSWeightedGreedy obtains the highest service-selection quality, 0.9063 ± 0.0027 , compared with 0.9001 ± 0.0021 for BS-DCSO. Higher local scores therefore do not necessarily imply higher complete-plan feasibility.

Feasible-plan utility also requires conditional interpretation because it is calculated only over successfully provisioned requests. QoSWeightedGreedy and ShortestLatency obtain mean FPU values of 0.6113 and 0.6202, respectively, compared with 0.6016 for BS-DCSO. However, ShortestLatency successfully provisions only approximately 3.1% of all requests, and its FPU is defined for only six seeds. A high conditional FPU for this small successful subset therefore does not indicate stronger overall provisioning capability.

The improved feasibility is obtained at substantially greater orchestration cost. BS-DCSO requires 168.19 ± 20.57 ms per request, whereas the greedy and latency-oriented methods are markedly faster. The comparison therefore reveals an explicit feasibility-computation trade-off introduced by Cartesian complete-plan evaluation over all resource nodes.

H. COMPLETE-CATALOGUE FEASIBILITY ANALYSIS

Across seeds 41–50, the complete-catalogue reference identifies 0.5867 ± 0.1214 of generated requests as hard-constraint feasible. Thus, approximately 58.7% of requests admit at least one feasible plan in the complete finite synthetic catalogue.

Conditional on this oracle-feasible set, BS-DCSO achieves an FISR of 0.8320 ± 0.0562 . Therefore, approximately 83.2% of requests known to contain at least one complete-catalogue feasible solution are successfully provisioned by BS-DCSO. Because BS-DCSO evaluates all resource nodes after service pruning, the remaining gap primarily reflects feasible functional-service alternatives removed by the per-type top-k restriction.

TABLE IV. Multi-seed feasibility and utility comparison. PSR denotes provisioning success rate, FISR denotes feasible-instance success rate, and FPU denotes feasible-plan utility.

Method	PSR	FISR	Violation	E2E Utility	FPU
BS-DCSO	0.4897 ± 0.1139	0.8320 ± 0.0562	0.0809 ± 0.0213	0.5850 ± 0.0095	0.6016 ± 0.0101
FeasibleShortestLatency	0.2710 ± 0.0662	0.4618 ± 0.0580	0.1619 ± 0.0322	0.5741 ± 0.0103	0.5987 ± 0.0125
GreedySemanticMatching	0.0593 ± 0.0642	0.1006 ± 0.0957	0.2509 ± 0.0431	0.5064 ± 0.0256	$0.5892 \pm 0.0283^\dagger$
QoSWeightedGreedy	0.2757 ± 0.1055	0.4447 ± 0.0996	0.1255 ± 0.0255	0.5702 ± 0.0070	0.6113 ± 0.0159
ResourceFirstScheduling	0.3473 ± 0.0715	0.6016 ± 0.0576	0.1073 ± 0.0161	0.5705 ± 0.0104	0.5785 ± 0.0110
ShortestLatency	0.0307 ± 0.0228	0.0612 ± 0.0489	0.2702 ± 0.0343	0.5709 ± 0.0169	$0.6202 \pm 0.0187^\ddagger$

TABLE V. Contextual relevance, local service-selection quality, and computation time in the multi-seed comparison.

Method	Semantic Relevance	Selection Quality	Runtime (ms/request)
BS-DCSO	0.2697 ± 0.0019	0.9001 ± 0.0021	168.19 ± 20.57
FeasibleShortestLatency	0.2502 ± 0.0015	0.8733 ± 0.0052	3.02 ± 0.50
GreedySemanticMatching	0.2871 ± 0.0018	0.8791 ± 0.0033	8.23 ± 0.02
QoSWeightedGreedy	0.2628 ± 0.0030	0.9063 ± 0.0027	22.02 ± 0.09
ResourceFirstScheduling	0.2435 ± 0.0040	0.8722 ± 0.0042	17.08 ± 1.81
ShortestLatency	0.2503 ± 0.0015	0.8752 ± 0.0061	1.31 ± 0.01

The oracle is used only as a feasibility certificate. It neither optimizes complete-plan utility nor establishes that the finite synthetic catalogue accurately represents the service availability, workload correlations, or infrastructure characteristics of a deployed smart-grid environment.

I. PAIRED SEED-LEVEL COMPARISON

Because all methods are evaluated on identical seed-specific benchmark instances, paired differences provide a more direct comparison than independently summarized absolute means. Table VI reports BS-DCSO minus baseline differences. For constraint violation, a negative value favors BS-DCSO; for PSR, FISR, and end-to-end utility, a positive value favors BS-DCSO.

For PSR, all five paired differences are positive and their 95% confidence intervals exclude zero. The smallest mean advantage occurs relative to ResourceFirstScheduling, for which BS-DCSO improves PSR by 0.1423 ± 0.0552 . Relative to FeasibleShortestLatency, the corresponding improvement is 0.2187 ± 0.0579 .

The same pattern is observed for FISR. The paired improvements range from 0.2304 ± 0.0701 relative to ResourceFirstScheduling to 0.7708 ± 0.0533 relative to ShortestLatency, with every reported confidence interval excluding zero.

For aggregate constraint violation, every paired difference is negative and every 95% confidence interval excludes zero. Thus, under the common seed-specific benchmark instances, BS-DCSO consistently returns plans with fewer hard-constraint violations than the controlled alternatives.

End-to-end utility exhibits a more qualified result. BS-DCSO has positive paired differences whose confidence intervals exclude zero relative to FeasibleShortestLatency, GreedySemanticMatching, QoSWeightedGreedy, and ResourceFirstScheduling. Relative to ShortestLatency, however, the difference is 0.0141 ± 0.0147 , whose confidence interval includes zero. The experiment therefore does not establish a consistent end-to-end utility improvement over

ShortestLatency, despite the substantially higher PSR and FISR of BS-DCSO.

Local metrics further demonstrate that BS-DCSO does not dominate every selection criterion. Relative to GreedySemanticMatching, its paired semantic-relevance difference is -0.0174 ± 0.0017 . Relative to QoSWeightedGreedy, its paired local service-selection quality difference is -0.0061 ± 0.0015 . These observations reinforce the interpretation that BS-DCSO is primarily a complete-plan feasibility method rather than a local semantic- or QoS-ranking optimizer.

J. FAILURE-RECOVERY ANALYSIS

The recovery experiment evaluates the re-orchestration mechanism described in Section 4. A failure specification is generated for every request using a configured failure rate of 1.0. The failed functional requirement is sampled uniformly from the concatenation of the request's required data and computing types. This sampling is uniform over functional requirements and therefore does not force equal numbers of data-service and computing-service failures.

For each evaluated method, the concrete service selected to implement the sampled functional requirement is identified and removed from the corresponding catalogue before re-orchestration. The same request-level failure specification is shared across methods and ablation variants. A recovery trial is considered eligible only when the original plan is fully feasible and the sampled functional requirement is implemented by that plan.

Under this protocol, the fault-injection eligible rate for BS-DCSO is numerically equal to its initial provisioning success rate, 0.4897 ± 0.1139 . Among eligible trials, BS-DCSO achieves a recovery success rate of 0.8282 ± 0.0398 , a conditional recovery utility of 0.6015 ± 0.0097 , and a mean re-orchestration time of 138.61 ± 29.99 ms. Table VII summarizes these results.

These results show that the proposed mechanism can frequently reconstruct a hard-constraint-feasible plan after

TABLE VI. Paired differences between BS-DCSO and the controlled baselines. Values are mean $\pm$ two-sided 95% Student's t confidence interval over ten paired seeds. Δ denotes BS-DCSO minus baseline.

Baseline	Δ PSR	Δ FISR	Δ Violation	Δ E2E Utility
FeasibleShortestLatency	+0.2187 $\pm$ 0.0579	+0.3702 $\pm$ 0.0558	-0.0810 $\pm$ 0.0234	+0.0109 $\pm$ 0.0059
GreedySemanticMatching	+0.4303 $\pm$ 0.1141	+0.7314 $\pm$ 0.1213	-0.1700 $\pm$ 0.0370	+0.0786 $\pm$ 0.0266
QoSWeightedGreedy	+0.2140 $\pm$ 0.0565	+0.3872 $\pm$ 0.1000	-0.0445 $\pm$ 0.0122	+0.0148 $\pm$ 0.0045
ResourceFirstScheduling	+0.1423 $\pm$ 0.0552	+0.2304 $\pm$ 0.0701	-0.0264 $\pm$ 0.0076	+0.0145 $\pm$ 0.0064
ShortestLatency	+0.4590 $\pm$ 0.1146	+0.7708 $\pm$ 0.0533	-0.1893 $\pm$ 0.0385	+0.0141 $\pm$ 0.0147

TABLE VII. BS-DCSO single-service failure-recovery results over seeds 41–50. Values are mean $\pm$ two-sided 95% Student's t confidence interval.

Metric	Mean $\pm$ 95% CI
Fault-injection eligible rate	0.4897 $\pm$ 0.1139
Recovery success rate	0.8282 $\pm$ 0.0398
Recovery utility	0.6015 $\pm$ 0.0097
Re-orchestration time (ms/eligible trial)	138.61 $\pm$ 29.99

the loss of one selected functional-service instance when suitable alternatives remain in the catalogue. The experiment does not evaluate simultaneous multi-service failures, resource-node failures, communication failures, regional outages, or cascading events and therefore should not be interpreted as a general resilience assessment of a deployed smart-grid platform.

K. ABLATION STUDY

The ablation experiment uses the same seeds 41–50 and the same seed-specific benchmark instances as the main comparison. Therefore, deterministic initial-provisioning metrics of the full BS-DCSO configuration match the corresponding main-experiment results. Runtime is obtained from separate script executions and consequently differs slightly from the timing values in Table V.

Four initial-provisioning variants are evaluated. Re-orchestration is not treated as an initial-provisioning ablation because it is inactive until a service failure occurs; its contribution is evaluated exclusively in Section 5.9.

Without semantic matching removes TF-IDF contextual relevance from complete-plan scoring while preserving the same top-k candidate space.

Without reliability scoring removes reliability from soft complete-plan scoring while preserving the hard minimum-reliability constraint.

Without cost-aware scoring removes cost satisfaction from soft complete-plan scoring while preserving the hard maximum-cost constraint.

Without resource-aware node selection evaluates a fixed catalogue resource node instead of all alternative nodes while preserving CPU, memory, and GPU capacity constraints. This variant removes resource-node search rather than resource feasibility.

Tables VIII and IX report absolute ablation results. Because all variants share the same seed-specific benchmark instances, paired full-minus-ablation differences provide the clearest measure of individual component effects. Tables X and XI report these comparisons.

The strongest ablation effect is associated with resource-aware node selection. Relative to the fixed-resource variant, full BS-DCSO improves PSR by 0.4613 ± 0.1291 and FISR by 0.7772 ± 0.1132 , reduces constraint violation by 0.0983 ± 0.0221 , and improves end-to-end utility by 0.0570 ± 0.0214 . All four confidence intervals exclude zero. Because CPU, memory, and GPU hard constraints remain active in the fixed-resource variant, this result specifically demonstrates the importance of searching alternative deployment nodes rather than merely enforcing resource constraints.

This benefit is accompanied by a substantial computational cost. The fixed-resource variant requires 17.51 ± 1.27 ms/request, compared with 164.36 ± 19.85 ms/request for full BS-DCSO in the ablation runs. The difference is consistent with the algorithmic design, because the full configuration evaluates every retained service composition on all available resource nodes.

Cost-aware scoring has a smaller but consistent effect. Relative to the no-cost variant, full BS-DCSO improves PSR by 0.0297 ± 0.0090 , FISR by 0.0546 ± 0.0181 , end-to-end utility by 0.0056 ± 0.0023 , and FPU by 0.0054 ± 0.0038 , while reducing constraint violation by 0.0049 ± 0.0015 . Each corresponding confidence interval excludes zero.

Reliability scoring exhibits a different pattern. Its removal does not yield a consistent change in PSR, FISR, or aggregate constraint violation: the corresponding paired differences are 0.0027 ± 0.0121 , 0.0071 ± 0.0233 , and -0.0003 ± 0.0021 , whose confidence intervals include zero. Nevertheless, the full configuration improves end-to-end utility by 0.0020 ± 0.0012 , FPU by 0.0024 ± 0.0014 , and local service-selection quality by 0.00499 ± 0.00098 . Reliability scoring therefore influences the composition and quality of selected plans more consistently than the probability of identifying a feasible plan in the current benchmark.

Removing TF-IDF semantic matching leaves PSR, FISR, and aggregate constraint violation exactly unchanged across the ten paired seeds. This result follows directly from the algorithm design because semantic similarity is excluded from top-k pruning and therefore does not alter the candidate space. Nevertheless, the full configuration improves semantic relevance by 0.00290 ± 0.00086 , end-to-end utility by 0.00041 ± 0.00013 , and FPU by 0.00043 ± 0.00019 . The full model also has a slightly lower local selection-quality score, with a paired difference of -0.00017 ± 0.00011 . These effects confirm that the TF-IDF component acts as a lightweight contextual preference among candidate plans rather than as a primary mechanism for improving feasibility.

TABLE VIII. Multi-seed ablation results over seeds 41–50. Values are mean $\pm$ two-sided 95% Student's t confidence interval.

Variant	PSR	FISR	Violation	E2E Utility	FPU
Full BS-DCSO	0.4897 $\pm$ 0.1139	0.8320 $\pm$ 0.0562	0.0809 $\pm$ 0.0213	0.5850 $\pm$ 0.0095	0.6016 $\pm$ 0.0101
Without resource-aware node selection	0.0283 $\pm$ 0.0430	0.0548 $\pm$ 0.0827	0.1793 $\pm$ 0.0175	0.5280 $\pm$ 0.0240	N/A [†]
Without cost-aware scoring	0.4600 $\pm$ 0.1128	0.7773 $\pm$ 0.0618	0.0858 $\pm$ 0.0211	0.5794 $\pm$ 0.0087	0.5962 $\pm$ 0.0094
Without reliability scoring	0.4870 $\pm$ 0.1124	0.8248 $\pm$ 0.0471	0.0812 $\pm$ 0.0217	0.5829 $\pm$ 0.0093	0.5993 $\pm$ 0.0102
Without semantic matching	0.4897 $\pm$ 0.1139	0.8320 $\pm$ 0.0562	0.0809 $\pm$ 0.0213	0.5846 $\pm$ 0.0095	0.6012 $\pm$ 0.0100

[†]Only two seeds contain valid FPU values for the fixed-resource variant. Consistent with the predefined reporting rule, FPU is reported as N/A rather than as a two-seed confidence interval.

TABLE IX. Contextual relevance, local service-selection quality, and runtime in the ablation experiment.

Variant	Semantic Relevance	Selection Quality	Runtime (ms/request)
Full BS-DCSO	0.2697 $\pm$ 0.0019	0.9001 $\pm$ 0.0021	164.36 $\pm$ 19.85
Without resource-aware node selection	0.2697 $\pm$ 0.0021	0.9000 $\pm$ 0.0026	17.51 $\pm$ 1.27
Without cost-aware scoring	0.2703 $\pm$ 0.0021	0.9005 $\pm$ 0.0022	162.79 $\pm$ 19.53
Without reliability scoring	0.2742 $\pm$ 0.0025	0.8951 $\pm$ 0.0020	162.97 $\pm$ 19.60
Without semantic matching	0.2668 $\pm$ 0.0026	0.9003 $\pm$ 0.0021	163.16 $\pm$ 19.65

TABLE X. Paired full-minus-ablation differences over seeds 41–50. A positive difference favors full BS-DCSO for PSR, FISR, E2E utility, and FPU, whereas a negative difference favors the full configuration for constraint violation.

Removed component	Δ PSR	Δ FISR	Δ Violation	Δ E2E Utility	Δ FPU
Resource-aware node selection	+0.4613 $\pm$ 0.1291	+0.7772 $\pm$ 0.1132	−0.0983 $\pm$ 0.0221	+0.0570 $\pm$ 0.0214	N/A [†]
Cost-aware scoring	+0.0297 $\pm$ 0.0090	+0.0546 $\pm$ 0.0181	−0.0049 $\pm$ 0.0015	+0.0056 $\pm$ 0.0023	+0.0054 $\pm$ 0.0038
Reliability scoring	+0.0027 $\pm$ 0.0121	+0.0071 $\pm$ 0.0233	−0.0003 $\pm$ 0.0021	+0.0020 $\pm$ 0.0012	+0.0024 $\pm$ 0.0014
Semantic matching	0.0000 $\pm$ 0.0000	0.0000 $\pm$ 0.0000	0.0000 $\pm$ 0.0000	+0.0004 $\pm$ 0.0001	+0.0004 $\pm$ 0.0002

[†]Only two paired seeds contain valid FPU values for the fixed-resource variant; the paired FPU difference is therefore reported as N/A.

TABLE XI. Paired effects of the ablations on semantic relevance and local service-selection quality. Values are full BS-DCSO minus the corresponding variant, reported as mean $\pm$ two-sided 95% Student's t confidence interval over ten paired seeds.

Removed component	Δ Semantic Relevance	Δ Selection Quality
Resource-aware node selection	−0.00002 $\pm$ 0.00020	+0.00013 $\pm$ 0.00207
Cost-aware scoring	−0.00056 $\pm$ 0.00087	−0.00038 $\pm$ 0.00068
Reliability scoring	−0.00445 $\pm$ 0.00174	+0.00499 $\pm$ 0.00098
Semantic matching	+0.00290 $\pm$ 0.00086	−0.00017 $\pm$ 0.00011

or general QoS.

L. SEQUENTIAL REQUEST-VOLUME TIMING ANALYSIS

The request-volume experiment examines how per-request orchestration time changes as the number of business requests increases while the generated service and resource catalogues remain fixed. Request volumes of 100, 300, 500, and 1000 are evaluated using benchmark seed 42. For each volume, one warm-up execution is followed by five timed repetitions. The measured interval covers orchestration only and excludes benchmark generation, TF-IDF fitting, complete-catalogue feasibility evaluation, and metric aggregation.

Across the four request-volume settings, mean BS-DCSO runtime ranges from 151.39 to 175.58 ms per request. Table XII summarizes the timing protocol and observed range.

The per-request runtime remains within the same order of magnitude as request volume increases from 100 to 1000 under the fixed-catalogue, sequential single-machine protocol. Within this sequential protocol, increasing the number of independently processed requests does not introduce an additional request-count-dependent search-space

TABLE XII. Sequential request-volume timing protocol and observed BS-DCSO runtime range.

Item	Value
Request volumes	100, 300, 500, and 1000
Benchmark seed	42
Warm-up runs per volume	1
Timed repetitions per volume	5
Measured runtime range (ms/request)	151.39–175.58
Timing scope	Orchestration only

expansion beyond repeated execution of the per-request orchestration procedure. However, the experiment does not evaluate concurrent arrivals, distributed execution, inter-request resource contention, or dynamically changing catalogues. It should therefore be interpreted as a descriptive request-volume timing experiment rather than evidence of high-concurrency scalability or suitability for protection and closed-loop control.

M. SUMMARY OF EXPERIMENTAL FINDINGS

Across the controlled multi-seed benchmark, BS-DCSO achieves the highest PSR and FISR and the lowest aggregate constraint violation among the evaluated orchestration

strategies. Paired comparisons against all five baselines favor BS-DCSO in PSR, FISR, and constraint violation, with the corresponding 95% confidence intervals excluding zero. The complete-catalogue reference identifies 0.5867 ± 0.1214 of requests as feasible, and BS-DCSO successfully provisions 0.8320 ± 0.0562 of these oracle-feasible instances, demonstrating that the bounded top-k search retains most, but not all, complete-catalogue feasibility.

The results also identify a clear feasibility–computation trade-off. Resource-aware evaluation across all deployment nodes is the dominant contributor to feasibility but accounts for a substantial portion of orchestration time. Cost-aware scoring provides a smaller but consistent contribution to feasibility and utility, while reliability scoring primarily refines selected-plan quality. TF-IDF contextual scoring does not affect initial feasibility because it is excluded from candidate pruning, but it produces small improvements in contextual relevance and complete-plan utility. Following a single selected functional-service failure, re-orchestration recovers a feasible replacement in 0.8282 ± 0.0398 of eligible trials.

These findings support BS-DCSO as a bounded complete-plan orchestration method for smart-grid operational analytics under the tested synthetic environment. They do not establish global optimality, superiority over published state-of-the-art service-composition algorithms, deployment-scale resilience, high-concurrency distributed performance, or suitability for hard real-time protection and closed-loop control.

VI. DISCUSSION

The experimental results indicate that the principal advantage of BS-DCSO lies in improving complete-plan feasibility under coupled functional, permission, quality, reliability, latency, cost, and resource-capacity constraints. BS-DCSO achieves a provisioning success rate of 0.4897 ± 0.1139 , compared with 0.3473 ± 0.0715 for ResourceFirstScheduling, the strongest controlled baseline in terms of absolute PSR. The paired BS-DCSO-minus-ResourceFirstScheduling difference is 0.1423 ± 0.0552 , and the corresponding paired PSR and FISR differences relative to all five controlled baselines have confidence intervals excluding zero. BS-DCSO also consistently reduces aggregate constraint violation. These findings suggest that evaluating service composition and resource placement jointly is more effective in the tested benchmark than optimizing either local service attributes or deployment decisions in isolation.

The results also show that stronger local ranking metrics do not necessarily translate into stronger end-to-end provisioning performance. GreedySemanticMatching achieves higher TF-IDF contextual relevance, while QoSWeightedGreedy achieves higher local service-selection quality. ShortestLatency attains a relatively high feasible-plan utility on the small subset of requests for which it succeeds, but its overall provisioning success is only approximately 3.1%. These results illustrate an important distinction between

local candidate quality and complete-plan executability. BS-DCSO does not attempt to maximize every local criterion; instead, it uses local ranking to restrict the candidate space and then evaluates complete plans under the full set of operational constraints.

The complete-catalogue feasibility analysis further clarifies the effect of bounded candidate pruning. The oracle feasibility rate is 0.5867 ± 0.1214 , whereas BS-DCSO achieves a feasible-instance success rate of 0.8320 ± 0.0562 . Because all resource nodes are evaluated after service pruning, the remaining gap between oracle feasibility and BS-DCSO feasibility is primarily associated with functional-service candidates removed by the per-type top-k restriction. This result demonstrates the intended trade-off of the proposed design: limiting the functional-service search substantially reduces the candidate space, but it can exclude service combinations that would otherwise satisfy all hard constraints. Accordingly, k_d and k_c should be interpreted as search-budget parameters rather than universally optimal values.

The ablation results show that the proposed components contribute in different ways. Resource-aware node selection has the largest effect on feasibility. Replacing all-node evaluation with a fixed resource node reduces PSR by 0.4613 ± 0.1291 and FISR by 0.7772 ± 0.1132 relative to the full configuration. Since CPU, memory, and GPU capacity constraints remain active in the fixed-resource variant, this difference reflects the benefit of searching alternative deployment nodes rather than simply enforcing resource constraints. In practical smart-grid data-fabric environments, this result suggests that service composition and resource placement should be considered jointly when edge and cloud nodes differ substantially in load and available capacity [38].

Cost-aware scoring produces smaller but statistically consistent improvements in feasibility and utility, indicating that soft cost preference can influence which retained plan eventually satisfies the complete set of constraints. Reliability scoring has a different role: removing it does not consistently alter PSR or FISR, but it reduces end-to-end utility, feasible-plan utility, and local selection quality. Reliability therefore acts primarily as a plan-refinement factor in the present benchmark rather than as a dominant determinant of search success.

The contribution of the TF-IDF component is comparatively modest and should be interpreted accordingly. Because contextual similarity is excluded from top-k candidate pruning, removing it leaves PSR, FISR, and aggregate constraint violation unchanged. Its effect appears only in complete-plan comparison, where it produces small but statistically consistent improvements in contextual relevance and utility. Thus, the term “business-semantic-aware” in BS-DCSO refers to lightweight contextual differentiation among functionally compatible candidate plans rather than ontology-level semantic reasoning. Richer power-domain representations, such as ontology-based concepts, knowledge graphs, or domain-specific embeddings, could provide stronger semantic discrimination, but their benefits would

need to be evaluated against their additional modeling and computational cost.

Failure-recovery results provide evidence that the same orchestration mechanism can also support service-level adaptation. After removal of one selected functional-service instance, BS-DCSO recovers a feasible replacement plan in 0.8282 ± 0.0398 of eligible trials. This indicates that alternative services and resource placements can often compensate for a single service loss when sufficient catalogue redundancy exists. However, the experiment considers only one failed functional-service instance at a time and therefore does not establish resilience against simultaneous service failures, resource-node failures, communication outages, regional disruptions, or cascading events.

The improved feasibility is accompanied by additional computational cost. BS-DCSO requires 168.19 ± 20.57 ms per request in the main multi-seed comparison, while request-volume measurements range from 151.39 to 175.58 ms per request for 100–1000 requests. The fixed-resource ablation is substantially faster, confirming that exhaustive evaluation across resource nodes accounts for a major portion of the computation. The current runtime is therefore best interpreted as a feasibility–computation trade-off. The method is suitable for investigation as an orchestration layer for operational analytics such as forecasting, equipment diagnosis, situational awareness, and maintenance-oriented applications, but the present measurements do not provide timing guarantees for protective relaying, emergency control, or other hard real-time closed-loop functions.

From an engineering perspective, BS-DCSO is positioned between business-level analytical requirements and heterogeneous smart-grid ICT resources. A practical deployment would require machine-readable catalogues describing functional service types, quality and reliability attributes, permissions, resource requirements, latency, cost, and current node availability [39]. Given these inputs, the algorithm determines which data and computing services should be combined and where the computing functions should execute. It therefore complements rather than replaces domain applications such as state estimation, load forecasting, fault diagnosis, or predictive maintenance: BS-DCSO provisions the required digital services and resources, whereas the downstream application remains responsible for electrical-system models and operational decisions.

Several limitations constrain the generalization of the present results. First, the benchmark is synthetic and does not reproduce all correlations, non-stationarity, communication delays, metadata errors, workload bursts, and resource dynamics of a deployed utility platform. Data-quality problems are also widely documented in other data-intensive domains [40]. Second, the controlled baselines isolate alternative decision paradigms but are not exact reproductions of published state-of-the-art service-composition or workflow-scheduling algorithms. Third, TF-IDF provides only lexical contextual matching and does not encode explicit power-

system ontology or topology relationships. Fourth, runtime measurements exclude service deployment, network transfer, analytical execution, and state migration, and the experiments do not evaluate concurrent request arrivals or distributed orchestration. Fifth, the complete-catalogue oracle certifies only the existence of a hard-constraint-feasible plan in the finite generated catalogue; it neither optimizes utility nor establishes global optimality of BS-DCSO. Finally, the recovery experiment considers only single functional-service failures. Broader fault-injection testing of service-oriented systems has been systematically investigated in the literature [41]. Future validation should therefore include real utility or open-grid service traces, stronger published baselines, exact optimal comparisons on tractable instances, richer power-domain semantic representations, adaptive candidate-pruning strategies, distributed online execution, and broader service, resource, and communication failure modes.

VII. PUBLICATION PRINCIPLES

This paper proposed BS-DCSO, a business-semantic-aware method for jointly orchestrating data services, computing services, and resource nodes in smart-grid data-fabric environments. The method combines exact functional matching, permission-aware candidate construction, QoS-based top-k pruning, lightweight TF-IDF contextual scoring, complete-plan evaluation across all available resource nodes, feasibility-first lexicographic selection, and failure-triggered re-orchestration. The resulting formulation explicitly separates hard executability from soft multi-attribute preference and treats service composition and resource placement as a coupled provisioning problem.

Across ten common random seeds, BS-DCSO achieves a provisioning success rate of 0.4897 ± 0.1139 , an oracle feasibility rate of 0.5867 ± 0.1214 , a feasible-instance success rate of 0.8320 ± 0.0562 , and an aggregate constraint violation rate of 0.0809 ± 0.0213 . Paired comparisons against all five controlled baselines favor BS-DCSO in provisioning success, feasible-instance success, and constraint violation. The ablation analysis shows that resource-aware node selection is the largest contributor to complete-plan feasibility, while cost and reliability scoring primarily refine plan selection and TF-IDF contextual scoring provides a smaller preference among otherwise compatible candidates. After a single selected functional-service failure, re-orchestration recovers a feasible replacement in 0.8282 ± 0.0398 of eligible trials.

These gains are obtained at the cost of additional computation because every retained functional-service composition is evaluated across all resource nodes. Under the tested synthetic setting, the present evidence supports BS-DCSO as a bounded feasibility-oriented orchestration method for smart-grid operational analytics rather than as a global optimizer, a high-concurrency scalability solution, or a hard real-time control mechanism. Future work will focus on validation using real utility and open-grid service data, comparison with published service-composition methods, richer

domain-specific semantic representations, adaptive pruning and distributed orchestration, and evaluation under larger dynamic workloads and more diverse failure conditions.

ACKNOWLEDGEMENTS

During the preparation of this manuscript, the authors used ChatGPT (OpenAI) for manuscript-wide language editing, restructuring, and presentation of author-provided experimental results. The tool was not used to generate experimental data, conduct experiments, calculate the reported results, or determine the scientific conclusions. All AI-assisted content was reviewed and verified by the authors, who take full responsibility for the final manuscript.

FUNDING

This work was supported by the Science and Technology Project of State Grid Jiangsu Electric Power Co., Ltd., “Research on Enhancing Full-Domain Information Network Support Capability and Key Technology Optimization” (Grant/Award No. J2024160).

REFERENCES

- [1] L. Xie, X. Zheng, Y. Sun, T. Huang, and T. Bruton, “Massively digitized power grid: Opportunities and challenges of use-inspired AI,” *Proc. IEEE*, vol. 111, no. 7, pp. 762–787, 2022.
- [2] M. Aghahadi, A. Bosisio, M. Merlo, A. Berizzi, A. Pegoianni, and S. Forciniti, “Digitalization processes in distribution grids: a comprehensive review of strategies and challenges,” *Appl. Sci.*, vol. 14, no. 11, Art. no. 4528, 2024.
- [3] G. Cheng, Y. Lin, A. Abur, A. Gómez-Expósito, and W. Wu, “A survey of power system state estimation using multiple data sources: PMUs, SCADA, AMI, and beyond,” *IEEE Trans. Smart Grid*, vol. 15, no. 1, pp. 1129–1151, 2023.
- [4] A. Bytyqi, S. Gandhi, E. Lambert, and N. Petrović, “A review on tso-dso data exchange, cim extensions and interoperability aspects,” *J. Mod. Power Syst. Clean Energy*, vol. 10, no. 2, pp. 309–315, 2022.
- [5] M. M. Thwe, A. Stefanov, V. S. Rajkumar, and P. Palensky, “Digital twins for power systems: Review of current practices, requirements, enabling technologies, data federation and challenges,” *IEEE Access*, 2025.
- [6] T. Rezník et al., “Improving the documentation and findability of data services and repositories: A review of (meta) data management approaches,” *Comput. Geosci.*, vol. 169, Art. no. 105194, 2022.
- [7] C. van Stiphoudt et al., “The Energy Synchronization Platform concept in the model region Augsburg to enable and streamline automated industrial demand response,” *Appl. Energy*, vol. 388, Art. no. 125455, 2025.
- [8] D. Davi-Arderius, E. Giovannetti, T. Jamasb, M. Llorca, and G. Soroush, “The Common European Energy Data Space and the Energy Networks,” *Curr. Sustainable/Renewable Energy Rep.*, vol. 13, no. 1, Art. no. 7, 2026.
- [9] P. Keshavarz Haddadha, M. H. Rezvani, M. MollaMotalebi, and A. Shankar, “Machine learning methods for service placement: a systematic review,” *Artif. Intell. Rev.*, vol. 57, no. 3, 2024. (no page numbers)
- [10] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi, “Resource scheduling in edge computing: A survey,” *IEEE Commun. Surv. Tut.*, vol. 23, no. 4, pp. 2131–2165, 2021.
- [11] N. Khaledian, M. Voelp, S. Azizi, and M. H. Shirvani, “AI-based & heuristic workflow scheduling in cloud and fog computing: a systematic review,” *Cluster Comput.*, vol. 27, no. 8, pp. 10265–10298, 2024.
- [12] T. Boroukhian, K. Supyen, C. W. McLaughlan, A. Bashyal, T. Pham, and H. Wicaksono, “Semantic middleware for demand response systems: Enhancing data interoperability in green electricity management for manufacturing,” *Comput. Ind.*, vol. 172, Art. no. 104354, 2025.
- [13] F. de Andrade Pereira et al., “A semantics-driven framework to enable demand flexibility control applications in real buildings,” *Adv. Eng. Inform.*, vol. 64, Art. no. 103049, 2025.
- [14] Q. Chen, Q. Li, J. Wu, C. Mao, G. Peng, and D. Wang, “Application of knowledge graph in power system fault diagnosis and disposal: A critical review and perspectives,” *Front. Energy Res.*, vol. 10, Art. no. 988280, 2022.
- [15] M. Razian, M. Fathian, R. Bahsoon, A. N. Toosi, and R. Buyya, “Service composition in dynamic environments: A systematic review and future directions,” *J. Syst. Softw.*, vol. 188, Art. no. 111290, 2022.
- [16] H. Taherdoost, “A systematic review of big data innovations in smart grids,” *Results Eng.*, vol. 22, Art. no. 102132, 2024.
- [17] E. Y. Song, G. J. FitzPatrick, K. B. Lee, and E. Griffor, “A methodology for modeling interoperability of smart sensors in smart grids,” *IEEE Trans. Smart Grid*, vol. 13, no. 1, pp. 555–563, 2021.
- [18] A. Gieß and A. Hutterer, “The Future of Data Management: A Delimitation of Data Platforms, Data Spaces, Data Meshes, and Data Fabrics,” *Inf. Syst. E-Bus. Manage.*, pp. 1–27, 2025.
- [19] A. Q. Al-Shetwi, I. E. Atawi, M. A. El-Hameed, and A. Abuelrub, “Digital Twin Technology for Renewable Energy, Smart Grids, Energy Storage and Vehicle-to-Grid Integration: Advancements, applications, key players, challenges and future perspectives in modernising sustainable grids,” *IET Smart Grid*, vol. 8, no. 1, Art. no. e70026, 2025.
- [20] A. Aghazadeh Ardebili, M. Zappatore, A. I. H. A. Ramadan, A. Longo, and A. Ficarella, “Digital Twins of smart energy systems: a systematic literature review on enablers, design, management and computational challenges,” *Energy Inform.*, vol. 7, no. 1, Art. no. 94, 2024.
- [21] N. Freitas, A. D. Rocha, and J. Barata, “Data management in industry: concepts, systematic review and future directions,” *J. Intell. Manuf.*, vol. 37, no. 2, pp. 799–827, 2026.
- [22] I. Blohm, F. Wortmann, C. Legner, and F. Köbler, “Data products, data mesh, and data fabric: I. Blohm et al.,” *Bus. Inf. Syst. Eng.*, vol. 66, no. 5, pp. 643–652, 2024.
- [23] Z. Gu et al., “A systematic overview of data federation systems,” *Semantic Web*, vol. 15, no. 1, pp. 107–165, 2024.
- [24] M. Masmoudi, S. Ben Abdallah Ben Lamine, M. H. Karray, B. Archimède, and H. Baazouai Zghal, “Semantic data integration and querying: A survey and challenges,” *ACM Comput. Surv.*, vol. 56, no. 8, pp. 1–35, 2024.
- [25] S. Hoseini, J. Theissen-Lipp, and C. Quix, “A survey on semantic data management as intersection of ontology-based data access, semantic modeling and data lakes,” *J. Web Semant.*, vol. 81, Art. no. 100819, 2024.
- [26] Y. M. Banad, S. S. Sharif, and Z. Rezaei, “Artificial intelligence and machine learning for smart grids: From foundational paradigms to emerging technologies with digital twin and large language model-driven intelligence,” *Energy Convers. Manage.*, vol. 28, Art. no. 101329, 2025.
- [27] N. Xiao, B. Peng, X. Li, J. Wu, J. Lou, and Y. Si, “Research on the construction and implementation of power grid fault handling knowledge graphs,” *Energy Rep.*, vol. 9, pp. 182–189, 2023.
- [28] B. Costa, J. Bachiega Jr, L. R. De Carvalho, and A. P. F. Araujo, “Orchestration in fog computing: A comprehensive survey,” *ACM Comput. Surv.*, vol. 55, no. 2, pp. 1–34, 2022.
- [29] S. R. Wilkinson et al., “Applying the FAIR principles to computational workflows,” *Sci. Data*, vol. 12, no. 1, Art. no. 328, 2025.
- [30] J. Lei, Q. Wu, and J. Xu, “Privacy and security-aware workflow scheduling in a hybrid cloud,” *Future Gener. Comput. Syst.*, vol. 131, pp. 269–278, 2022.
- [31] J. Powell, A. McCafferty-Leroux, W. Hilal, and S. A. Gadsden, “Smart grids: A comprehensive survey of challenges, industry applications, and future trends,” *Energy Rep.*, vol. 11, pp. 5760–5785, 2024.
- [32] A. Goknil et al., “A systematic review of data quality in CPS and IoT for industry 4.0,” *ACM Comput. Surv.*, vol. 55, no. 14s, pp. 1–38, 2023.
- [33] N. Soveizi, F. Turkmen, and D. Karastoyanova, “Security and privacy concerns in cloud-based scientific and business workflows: A systematic review,” *Future Gener. Comput. Syst.*, vol. 148, pp. 184–200, 2023.
- [34] T. Chen and M. Li, “The weights can be harmful: Pareto search versus weighted search in multi-objective search-based software engineering,” *ACM Trans. Softw. Eng. Methodol.*, vol. 32, no. 1, pp. 1–40, 2023.
- [35] M. R. Shadi, H. Mirshekari, and H. R. Shaker, “Explainable artificial intelligence for energy systems maintenance: A review on concepts, current techniques, challenges, and prospects,” *Renew. Sustain. Energy Rev.*, vol. 216, Art. no. 115668, 2025.
- [36] D. J. Eckman, S. G. Henderson, and S. Shashaani, “SimOpt: A testbed for simulation-optimization experiments,” *INFORMS J. Comput.*, vol. 35, no. 2, pp. 495–508, 2023.
- [37] H. Semmelrock et al., “Reproducibility in machine-learning-based research: Overview, barriers, and drivers,” *AI Mag.*, vol. 46, no. 2, Art. no. e70002, 2025.
- [38] F. U. Khan, I. A. Shah, S. Jan, S. Ahmad, and T. Whangbo, “Machine learning-based resource management in fog computing: A systematic literature review,” *Sensors*, vol. 25, no. 3, Art. no. 687, 2025.
- [39] A. V. Tokmak, A. Akbulut, and C. Catal, “Web service discovery: Rationale, challenges, and solution directions,” *Comput. Stand. Interfaces*, vol. 88, Art. no. 103794, 2024.
- [40] R. Syed et al., “Digital health data quality issues: systematic review,” *J. Med. Internet Res.*, vol. 25, Art. no. e42615, 2023.
- [41] S. Yu, H. Wu, X. Niu, and C. Nie, “A Systematic Literature Review on Fault Injection Testing of Microservice Systems,” *IEEE Trans. Services Comput.*, 2025.