Received: 27 June 2022 | Revised: 3 November 2022 | Accepted: 27 November 2022

DOI: 10.1002/aic.17977

RESEARCH ARTICLE

Thermodynamics and Molecular-Scale Phenomena

AIChE JOURNAL

Data-driven coordination of subproblems in enterprise-wide optimization under organizational considerations

Damien van de Berg | Panagiotis Petsagkourakis | Nilay Shah | Ehecatl Antonio del Rio-Chanona

Sargent Centre for Process Systems Engineering, Roderic Hill Building South Kensington Campus London, London, UK

Correspondence
Nilay Shah and Ehecatl Antonio del Rio-Chanona, Sargent Centre for Process Systems Engineering, Roderic Hill Building South Kensington Campus London, London SW7 2AZ, UK.
Email: n.shah@imperial.ac.uk and a.del-rio-chanona@imperial.ac.uk

Abstract

While decomposition techniques in mathematical programming are usually designed for numerical efficiency, coordination problems within enterprise-wide optimization are often limited by organizational rather than numerical considerations. We propose a "data-driven" coordination framework which manages to recover the same optimum as the equivalent centralized formulation while allowing coordinating agents to retain autonomy, privacy, and flexibility over their own objectives, constraints, and variables. This approach updates the coordinated, or shared, variables based on derivative-free optimization (DFO) using only coordinated variables to agent-level optimal subproblem evaluation "data." We compare the performance of our framework using different DFO solvers (CUATRO, Py-BOBYQA, DIRECT-L, GPyOpt) against conventional distributed optimization (ADMM) on three case studies: collaborative learning, facility location, and multiobjective blending. We show that in low-dimensional and nonconvex subproblems, the exploration-exploitation trade-offs of DFO solvers can be leveraged to converge faster and to a better solution than in distributed optimization.

KEYWORDS
coordination, data-driven optimization, distributed optimization, expensive black-box

1 | INTRODUCTION

Companies within the process industries rely on mathematical optimization for their operations to remain competitive in an environment of increasingly stringent safety, environmental, and economic requirements.[1] This gives rise to the field of enterprise-wide optimization (EWO) with the ultimate goal to coordinate all decision-making within a company.[2,3] EWO involves the integration of units across (1) all hierarchical levels of decision-making (from design, planning, scheduling, to control), and (2) all geographically distributed (plants, warehouses, etc.) or functional (sourcing, manufacturing, distribution) units. Conventionally, these separate entities are solved sequentially via one-way information flow.[4] For instance, higher-level planning might determine the setpoints of lower-level scheduling without explicitly accounting for lower-level constraints; or geographically separated plants might adjust their operations to accommodate the needs of other bottleneck plants in the value chain. These heuristics in coordinated decision-making, while sometimes necessary for practicality and tractability, do not guarantee optimality of the integrated problem. However, integrated model-based optimization traditionally requires the solution of a larger-scale centralized optimization model, which quickly becomes computationally intractable in the number of decision variables and constraints.[4] A centralized formulation could also in practice be obviated by organizational complexity (antitrust, privacy, and more).

One way to alleviate the computational burden of model-based integration is to relax constraints, or replace detailed formulations with surrogate models that are easier to handle by numerical solvers.[5–7] Usually, this would come at the expense of a degradation in solution quality. However, since EWO aims to coordinate previously decoupled decision-making, the resulting optimization formulations present mathematical structures that can be exploited. The resulting problems comprise few complicating variables and constraints that lend themselves well to decomposition and distributed optimization schemes.[8] Decomposition techniques consist of the iterative solution of a relaxed upper- and reduced-order lower-level problem which can theoretically achieve the same solution quality as the original formulation, while saving computational time. Bilevel[9,10] and Benders decomposition[11–13] are among the most prominent techniques for addressing complicating variables, and typically decompose problems over time, and stochastic realizations of uncertainty respectively. Lagrangean decomposition is particularly useful for tackling complicating constraints, as well as complicating variables by reformulation. As such, it is also useful for decomposing problems by time, space, or products.[14–17]

Distributed optimization builds on the concepts of dual decomposition techniques (such as Lagrangean decomposition). It has many applications in problems that are separated by complicating constraints, such as the integration of geographically dispersed warehouses or plants along a supply chain.[18] The Alternating Direction Method of Multipliers (ADMM) has received special attention as a powerful tool enabling considerable computational savings using minimal information exchange, especially in convex optimization.[19] ADMM repeatedly iterates between the solution of private, localized, lower-level subproblems, and an upper-level problem whose aim is to coordinate the solutions of the private subproblems. The possibility for solving the subproblems in parallel gives rise to significant potential computational savings. Despite often being applied in practice, ADMM loses its convergence guarantees on nonconvex problems.[20] Another drawback of ADMM is that the method practically only leads to computational savings compared to the *centralized* solution under special conditions, namely when the problem is decomposed into *numerous*, *convex* subproblems.[19]

Similarly to how the convergence of first-order gradient descent solvers can be improved using acceleration or momentum, there are several ways to speed up the convergence of ADMM using similar schemes.[21] Houska et al.[22] have proposed ALADIN, an algorithm to address ADMM's shortcomings: it speeds up—and includes theoretical conditions for—global convergence to local minimizers on nonconvex problems. ALADIN iterates between the parallel optimization of subproblems and sequential quadratic programming (SQP) steps for the coordination around the local subproblem solutions.

While distributed optimization seems promising from a computational perspective, much of the literature discussing model-based integration in EWO with relevant solution techniques fails to consider communication and business considerations that could hinder their practical applicability.[23–25] Distributed optimization is often approached using a top-down coordination approach: Starting from a centralized model, a decomposition is applied that is expected to lead to computational savings. This presupposes that previously decoupled decision-makers (1) are willing to share their local models; (2) accept the risk of foregoing a certain degree of autonomy, flexibility, and Nash equilibria for the pursuit of the "social optimum" of the centralized model; and (3) even have access to known, differentiable expressions as part of their optimization model. Due to a significant increase in computational power over the past few decades, software and organizational rather than numerical considerations might become the bottleneck in the integration of computational decision-making.[26] In fact, current decision-making architectures were often established within a legal and organizational framework when operations were (and often still are) guided by heuristics rather than numerical optimization. As such, the considered problem is rendered into a multiagent coordination problem where each agent might represent a separate legal entity with its own autonomy, agenda, technical constraints, and organizational considerations.[27,28]

The organizational context matters when choosing the best coordination scheme. When all agents are willing to collaborate and share differentiable model expressions, powerful distributed optimization techniques can be leveraged for optimal numerical efficiency.[29] When coordinating (not necessarily collaborating) agents only have access to black-box simulation tools for decision-making, "data-driven" or "black-box" optimization tools need to be adapted for the coordination. There are many reviews on data-driven or derivative-free optimization algorithms.[30,31] Some state-of-the-art methods have also been benchmarked on typical process systems engineering (PSE) applications in Reference 32, and have been introduced to solve multilevel problems in References 33–35. van de Berg et al.[36] show that derivative-free optimization can be used for the data-driven coordination of black-box subproblems in multiobjective problems arising in PSE.

In this work, we build upon van de Berg et al.[36] to investigate whether derivative-free optimization (DFO) can be used as a viable alternative to distributed optimization solvers in the following coordination problems: each agent is willing to collaborate (i.e., sacrifice suboptimality in their own objective for a "greater good") and has their own decision-making model, which does *not* have to be white-box—it could be the black-box result of a third-party, proprietary simulation software. In the context of EWO, this problem might arise when plants along the same value chain need to coordinate on material streams given that each plant has a separate objective that they optimize with the help of third-party software. In this case, the model is not readily exploitable for gradient information, such that solvers like ALADIN cannot be used as it requires exact first-order gradient information of the subproblems for its SQP step. The question arises if data-driven optimization approaches perform best for these kinds of scenarios.

While the performance of different distributed optimization algorithms have been compared with each other and with a centralized solution,[37] we thoroughly investigate under which conditions data-driven optimization outperforms typical distributed optimization solvers such as ADMM. As discussed in van de Berg et al.,[36] any (potentially imperfect) gradient information becomes increasingly valuable in higher-dimensional decision spaces. Since ADMM's upper-level coordination step involves subgradient information, we only expect DFO to be competitive under specific conditions, that is, when

the number of complicating variables is few relative to the number of private decision variables. Our aim is not to outperform centralized solution methods. In the methods we compare, computational efficiency is sacrificed for agent privacy, autonomy, and flexibility.

This article is organized as follows: In Section 2, we illustrate our data-driven methodology along with conventional ADMM. We also explain our choice of DFO algorithms (CUATRO, Py-BOBYQA, DIRECT-L, and GPyOpt) for our data-driven coordination problems. In Section 3, we then introduce a motivating mathematical test function and three case studies. In Section 4, we then present and discuss the convergence of all algorithms. We also investigate how algorithm convergence changes with the number of complicating variables, the number of coordinating agents, and the topology of the subproblem solution space.

2 | METHODOLOGY

2.1 | Problem statement

We are interested in solving the equivalent of the following centralized, integrated coordination problem:

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{X}} \quad & \sum_{i=1}^{N} f_i(\mathbf{x}_{i,p}, \mathbf{z}) \\ \text{s.t.} \quad & \mathbf{g}_i(\mathbf{x}_{i,p}, \mathbf{z}) \le \mathbf{0}, \;\; i = 1...N \end{aligned} \tag{1}$$

where $\mathbf{x} \in \mathbb{X} \subset \mathbb{R}^{n_x}$ refers to the decision variable vector within feasibility set $\mathbb{X}$. As such, $\mathbf{x}$ includes not only the "local," private decision variables $\mathbf{x}_{i,p} \in \mathbb{X}_{i,p} \subset \mathbb{R}^{n_{x_i}}$ of all N agents, but also the "global," shared variables $\mathbf{z}$ within the feasibility set $\mathcal{Z} \subset \mathbb{R}^{n_z}$. As such, the complete decision vector comprises the following elements: $\mathbf{x} = [\mathbf{x}_{1,p}, ..., \mathbf{x}_{N,p}, \mathbf{z}]$. The optimization is also subject to N local agent constraints $\mathbf{g}_i : \mathbb{R}^{n_{x_i}} \times \mathbb{R}^{n_z} \to \mathbb{R}^{n_{g_i}}$.

This generic problem formulation also implicitly allows for the inclusion of equality constraints in Equation (1) through a reduction in the degrees of freedom of the decision variables, or through an equivalent reformulation into two inequalities. Additionally, Equation (1) also allows for the incorporation of global, or shared, constraints and objectives. We would call any constraint g_{global} "shared" if it only depends on the shared variables $\mathbf{z}$. Similarly, shared objective terms might either manifest as a separate term f_{global} in a single agent objective, or be incorporated into the objectives of any $M \le N$ agents as $\frac{f_{\text{global}}(\mathbf{z})}{M}$.

2.2 | Problem reformulation

Problem (1) can be reformulated into:

$$\begin{aligned} \min_{\mathbf{z}\in\mathcal{Z}} \quad & \min_{\mathbf{x}_{i,p}, i=1,...,N} \quad \sum_{i=1}^{N} f_i(\mathbf{x}_{i,p}, \mathbf{z}) \\ & \text{s.t.} \qquad \mathbf{g}_i(\mathbf{x}_{i,p}, \mathbf{z}) \le \mathbf{0}, \;\; i = 1...N \end{aligned} \tag{2}$$

After fixing $\mathbf{z}$, the problem becomes block separable, which makes the problem amenable to decomposition and distributed optimization. This becomes evident when rewriting Equation (2) as its equivalent constrained (bilevel) optimization problem in (3). The coordination step involves an update in the shared variables $\mathbf{z}$. At each iteration, the subproblems are solved *in private* to find the optimal objective $F_i(\mathbf{z})$ and set of private variables $\mathbf{x}^*_{i,p}$ corresponding to a set of shared variables $\mathbf{z}$. Agents can maintain autonomy and flexibility by deciding on their own objective and constraint functions which they do not need to share with other agents. The only information that agents share with a third-party coordinator is the optimal set of private variables and local copy of shared variables $\mathbf{x}^*_{i,p}$ and $\mathbf{z}^*_i$ (2.3) *or* the optimal objective $f^*(\cdot)$ (2.4) corresponding to a suggested set of shared variables $\mathbf{z}$. For simplicity's sake, we assume that the subproblems are solved to global optimality. This is a necessary condition to find the globally optimal proposed set of shared variables in the upper level. In practice however, few industrial problems are provably solved to global optimality and we could still find a "good" $\mathbf{z}$ in both frameworks if this condition is not met. The bottleneck in achieving global optimality in our considered case studies is usually the coordination step rather than violation of this assumption. The global optimality convergence considerations in ADMM and the suggested framework are addressed in further detail in Sections 2.3 and 2.4. On top of this, we do not assume that the lower-level problems have to be solved by exploiting known expressions. In fact, the subproblem optimization could involve black-box queries such as proprietary simulations.

$$\begin{aligned} \min_{\mathbf{z}\in\mathcal{Z}} \quad & F(\mathbf{z}) \\ \text{s.t.} \quad & F(\mathbf{z}) = \sum_{i}^{N} F_i(\mathbf{z}), \;\; \text{where } \forall i \in \{1, ..., N\}: \\ & F_i(\mathbf{z}) = \min_{\mathbf{x}_{i,p}\in\mathbb{X}_i} \; f_i(\mathbf{x}_{i,p}, \mathbf{z}) \\ & \qquad \text{s.t.} \quad \mathbf{g}_i(\mathbf{x}_{i,p}, \mathbf{z}) \le \mathbf{0} \end{aligned} \tag{3}$$

2.3 | ADMM by consensus

The conventional method that our proposed approach is benchmarked against is ADMM in its consensus form, which is presented in Algorithm 1 and in more detail in Appendix (D.4). After initialization (**step 1**), ADMM iterates over **steps 2–7** until the evaluation budget is exhausted: This involves the solution of subproblems in private and parallel (**steps 3–5**) to get the local copy of shared variables $\mathbf{z}_i$, and an update in the shared variables $\mathbf{z}$ and scaled dual variables $\mathbf{u}_i$ based on $\mathbf{z}_i$ (**step 6**).

In **step 4**, each agent optimizes their copy of shared/complicating variables $\mathbf{z}_i$ that minimizes their private objective function while penalizing any deviation from the *suggested* value of the complicating variables $\mathbf{z}^k$:

$$\begin{aligned} \mathbf{x}^{k+1}_{i,p}, \mathbf{z}^{k+1}_i \;\leftarrow\; & F_i(\mathbf{z}^k) \\ = \; & \arg\min_{\mathbf{x}_{i,p}, \mathbf{z}_i} f_i(\mathbf{x}_{i,p}, \mathbf{z}_i) + \frac{\rho}{2}\left\|\mathbf{z}_i - \mathbf{z}^k + \mathbf{u}^k_i\right\|^2_2 \;\; \text{s.t.} \;\; \mathbf{g}_i(\mathbf{x}_{i,p}, \mathbf{z}_i) \le \mathbf{0} \end{aligned} \tag{4}$$

ALGORITHM 1 **Alternating Direction Method of Multipliers (ADMM) by consensus**

Input: Agent objectives f_i and constraints $g_{i,k}, k = 1 \dots n_{g_i}, i = 1 \dots N$, Initial shared variable guess $\mathbf{z}^0$, Penalty parameter ρ, Maximum number of function evaluations $N_{f,max}$

1 **Initialisation:** Initial dual variables $\mathbf{u}_i^0 = [0, \dots, 0]$

2 **for** $j = 0 \dots N_{f,max} - 1$ **do**

3 **for** *agent* $i = 1 \dots N$ *in parallel* **do**

4 $\mathbf{x}_{i,p}^{k+1}, \mathbf{z}_i^{k+1} \leftarrow F_i(\mathbf{z}^k)$, by solving lower-level problem (4)

5 **end**

6 $\mathbf{z}^{k+1} \leftarrow \frac{1}{N} \sum_{i=1}^{N} \mathbf{z}_i^{k+1}, \quad \mathbf{u}_i^{k+1} \leftarrow \mathbf{u}_i^k + \mathbf{z}_i^{k+1} - \mathbf{z}^{k+1}$;

7 **end**

where $\|\cdot\|_2$ refers to the Frobenius norm, and $\mathbf{u}_i$ to the scaled dual variables of agent i. $\frac{\rho}{2}\|\mathbf{z}_i - \mathbf{z}^k + \mathbf{u}_i^k\|_2^2$ is known as the proximal or penalty term and is useful for stabilizing convergence in the shared variables $\mathbf{z}$. It also makes the formulation robust against local constraints: when there is no feasible set of $\mathbf{x}_{i,p}$ that satisfy all constraints for a given $\mathbf{z}^k$, the solution converges to the nearest feasible $\mathbf{z}_i$ incurring a penalization in the objective. After all subproblems are solved, the set of suggested complicating variables is then updated to $\mathbf{z}^{k+1}$ in the coordination step (**step 6**) by averaging the set of optimal complicating variables resulting from the agent subproblems $\mathbf{z}_i^k$. While the update in the shared variables $\mathbf{z}^{k+1}$ aims to ensure asymptotic primal feasibility, the update in the dual variables $\mathbf{u}_i^{k+1}$ aims to ensure asymptotic dual feasibility. Each agent's dual variables are updated to $\mathbf{u}_i^{k+1}$ based on the difference between $\mathbf{u}_i^k$ and the local copy of shared variables $\mathbf{z}_i^{k+1}$, and could be interpreted as an integral error term often encountered in control.

A common drawback of ADMM is that it can take many iterations to converge to a high-accuracy solution and that it is in theory only proven to converge to the global optimum when the original formulation (and thus the subproblems) is convex.[21,38,39] This begs the question if the coordination step in $\mathbf{z}$ could be improved to speed up the convergence or find a better solution quality for a given evaluation budget.

2.4 | Data-driven coordination

Problem (3) views the coordination formulation as a bilevel optimization instance. Derivative-free optimization (DFO) has already been used to solve for the upper-level variables in multilevel problems,[33–35] and as such presents a promising alternative to ADMM's subgradient update step. The difference between the data-driven coordination framework and ADMM is illustrated in Figure 1 and the data-driven coordination framework is illustrated in more detail in Algorithm 2: After initialization (**step 1**), our framework iterates over **steps 2–11** until the evaluation budget is exhausted: In **step 3**, the upper level aims to find the set of complicating variables that minimize the objective function *subject to* the *optimal* solution in parallel of the agent-level subproblems in the private variables (**steps 4–9**).

Step 3 uses a DFO algorithm to update the shared variables $\mathbf{z}$ with the aim to minimize the "black-box" upper-level objective $F(z)$ in Equation (3).

$$\min_{\mathbf{z} \in \mathcal{Z}} \; F(\mathbf{z}) \tag{5}$$

where the decision variables $\mathbf{z}$ are subject to box-bound constraints $\mathcal{Z}$. Any box-constrained derivative-free, black-box, data-driven, or "zeroth-order" optimization algorithm can be used for the solution of the upper level.[30–32] Since the "black-box evaluations" are the result of optimizations, these evaluations are considered expensive. The number of evaluations n_{next} that are sampled at each iteration in **step 3** depends on the exploitation-exploration trade-off as well as sampling strategy of the DFO method used.

$F(\mathbf{z})$ is obtained in **steps 4–9** in a similar manner to Equation (3). $F_i(\mathbf{z})$ is treated as the result of private black-box simulations and $F(\mathbf{z})$ is equivalent to the sum of all optimal subproblem solutions in Equation (4), with the exception that the objective omits any dual variables:

$$\begin{aligned} F(\mathbf{z}) = \sum_i^N F_i(\mathbf{z}) \;\; \text{where} \;\; F_i(\mathbf{z}) = \min_{\mathbf{x}_{i,p}, \mathbf{z}_i} . \, & f_i(\mathbf{x}_{i,p}, \mathbf{z}_i) \\ + \frac{\rho}{2}\|\mathbf{z}_i - \mathbf{z}^k\|_2^2 \;\; \text{s.t.} \;\; & \mathbf{g}_i(\mathbf{x}_{i,p}, \mathbf{z}_i) \le \mathbf{0} \end{aligned} \tag{6}$$

The scaled dual is omitted as it only enhances convergence within the rigorous stability scheme of ADMM,[40] and can even degrade

convergence performance. The reason we still need to introduce the proximal term and the local copies $\mathbf{z}_i$ to be used instead of $\mathbf{z}^k$ in $f_i(\mathbf{x}_{i,p}, \mathbf{z}_i)$ and $\mathbf{g}_i(\mathbf{x}_{i,p}, \mathbf{z}_i)$ is the same as for ADMM: $\mathbf{g}_i(\mathbf{x}_{i,p}, \mathbf{z}^k)$ might not be feasible for the current iteration which would lead to a failure of the subproblem. As such, the go-to Formulation we use in our proposed methodology is (6). However, if we know that our initial guess of $\mathbf{z}$ is feasible, we can use constraint-handling DFO solvers that can use the binary subproblem feasibility information to explore the solution space. This is discussed in more detail in Section 2.4.2.

When the subproblems are solved to global optimality, the whole optimization problem can be solved (heuristically or rigorously) to global optimality depending on the function evaluation budget and the convergence certificate of the DFO solver. Among the proposed DFO solvers, only DIRECT-L has this asymptotic global convergence property when no assumptions on the subproblem convexity is made. Since convergence is limited by the number of expensive subproblem calls, we do not include overly exploratory methods, such as particle swarm methods. In the next section, we explore any analogies to "data-driven" ADMM and ALADIN when quadratic surrogates (CUATRO) are used for the DFO step. Additionally, we introduce the other DFO algorithms used, whose choice is informed by van de Berg et al.[32]: Py-BOBYQA as the trust region model-based method, DIRECT-L as the direct method, and GPyOpt for Bayesian Optimization. Figure 2 shows our selection of data-driven as well as distributed optimization algorithms, and their mutual relations.

2.4.1 | Data-driven distributed optimization

DFO algorithms can be classified into direct and model-based methods. Direct methods optimize an expression by directly handling function evaluations, while model-based methods rely on the intermediate construction and optimization of surrogates.[30] While Formulation (6) is applicable for both direct and model-based DFO solvers,

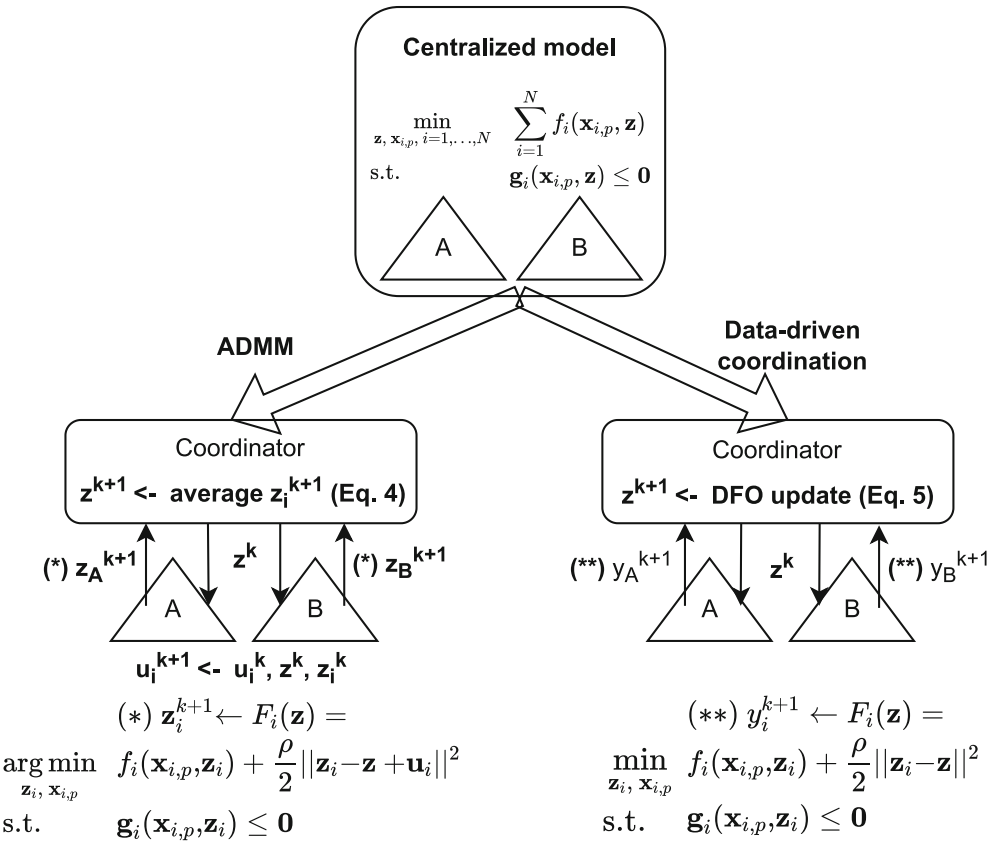

FIGURE 1 We can solve an equivalent centralized formulation either by ADMM or data-driven coordination (⇓). In either case, a coordinator (□) iteratively sends an updated proposed set of shared variables $\mathbf{z}^k$ (↓) to all coordinating agents. The coordinating agents (Δ) then optimize their private objective according to (*) or (**) for ADMM and data-driven coordination respectively. The subproblems differ in whether they include the scaled dual variables $\mathbf{u}_i^{k+1}$—updated in the preceding step for ADMM—in their construction of the proximal term. Additionally, in ADMM, the subproblems return the optimal local copies of the proposed shared variables $\mathbf{z}_i^{k+1}$, whereas our framework returns the optimal subproblem objective evaluations y_i^{k+1} (↑). In both frameworks, the penalty parameter ρ determines the extent to which the deviation between the suggested $\mathbf{z}^k$ and optimal set of private variables $\mathbf{z}_i^{k+1}$ is penalized. In the last step of the iteration, ADMM updates the proposed set of shared variables $\mathbf{z}^{k+1}$ by averaging its local copies $\mathbf{z}_i^{k+1}$, while our data-driven framework updates $\mathbf{z}^{k+1}$ using derivative-free optimization (DFO) and shared variable $\mathbf{z}^{k+1}$ to optimal evaluation y_i^{k+1} input–output data. In this case, we only show two coordinating agents, but this scheme can be generalized to any number of coordinating agents.

ALGORITHM 2 **Data-driven coordination framework**

Input: Agent objectives f_i and constraints $g_{i,k}, k = 1 \dots n_{g_i}, i = 1 \dots N$, Box-bound constraints $\mathcal{Z} \in \mathbb{R}^{n_z \times 2}$ on $\mathbf{z}$, Initial shared variable guess $\mathbf{z}^0$, Penalty parameter ρ, Maximum number of function evaluations $N_{f,max}$

Initialisation: Function evaluation counter $n_f := 1, \mathbf{z}_{best} = \mathbf{z}^0, y_{best} = F(\mathbf{z}^0)$ where $F(\cdot)$ is obtained from (6), initial data sets ($Z := \{\mathbf{z}^0\}, \mathbf{y} := \{y_{best}\}$

while $(N_{f,max} \geq n_f)$ **do**

  Obtain n_{next} samples Z_{next} from DFO update step at $\mathbf{z}_{best}$ using (5) ;

  for $\mathbf{z}_j = \mathbf{z}_1 \dots \mathbf{z}_{next} \in Z_{next}$ **do**

    for *agent* $i = 1 \dots N$ *in parallel* **do**

      Obtain $F_i(\mathbf{z}_j)$ from (6)

    end

    Update datasets: $Z \leftarrow \{Z, \mathbf{z}_j\}, \mathbf{y} \leftarrow \{\mathbf{y}, y_{next}\}$, where $y_{next} = \sum_i^N F_i(\mathbf{z}_j)$

  end

  Update best iterate: $n_f \leftarrow n_f + n_{next}, \quad y_{best} \leftarrow \min_{j=1\dots n_f} . \mathbf{y}_j, \quad \mathbf{z}_{best} \leftarrow \arg\min_{j=1\dots n_f} . y_j$;

end

model-based DFO methods can address Problem (5) by introducing surrogates $\widehat{F}(\mathbf{z})$ in two different ways: One option would be to fit a single surrogate over the sum of the subproblem evaluations, that is, the optimal value of the subproblem objectives for a given proposed set of shared variables $\mathbf{z}$.

$$\min_{\mathbf{z} \in \mathcal{Z}} \ \widehat{F}(\mathbf{z}) \ \text{ where } \ \widehat{F}(\mathbf{z}) \approx \sum_i^N F_i(\mathbf{z}) \tag{7}$$

A second alternative would be to allow for separate surrogates to be fitted for each subproblem before the sum of surrogates is optimized in the upper level.

$$\min_{\mathbf{z} \in \mathcal{Z}} \ \sum_i^N \widehat{F}_i(\mathbf{z}) \ \text{ where } \ \widehat{F}_i(\mathbf{z}) \approx F_i(\mathbf{z}) \tag{8}$$

Similar to van de Berg et al.,[36] we use convex quadratic surrogates ($\widehat{F}(\mathbf{z}) = \mathbf{z}^\top A\mathbf{z} + \mathbf{b}^\top \mathbf{z} + c, \ A \succeq 0 \in \mathbb{R}^{n_z \times n_z}, \mathbf{b} \in \mathbb{R}^{n_z \times 1}, c \in \mathbb{R}$) within the CUATRO framework. In this case, the approach used in Equation (7) is similar to,[41] and could be loosely referred to as "Data-driven ADMM." The approach in (8) could then be viewed as "Data-driven ALADIN," with a crucial difference: ALADIN's quadratic surrogate coefficients are given by the gradient and Hessian (obtained via automatic differentiation) of a second-order Taylor expansion around the local subproblem solutions, while in our data-driven counterpart, the surrogates are obtained via quadratic regression based on the subproblem evaluations. Whenever possible, we should leverage automatic differentiation to obtain gradient information with respect to the shared variables as this is expected to yield better convergence with a higher accuracy especially in higher dimensions. In this work however, we cannot use ALADIN or automatic differentiation as this makes the comparison to the subproblem-agnostic ADMM and proposed approach unfair since we are assuming that the subproblem expressions are not necessarily known.

2.4.2 | CUATRO

We modified CUATRO—a quadratic trust-region surrogate-based DFO algorithm—to be used within the "data-driven ADMM" (ADMM_CUATRO) and "data-driven ALADIN" (ALADIN_CUATRO) framework. The reader is referred to our previous work van de Berg et al.[32] for a detailed description of the algorithm. CUATRO is chosen as our quadratic surrogate-based DFO algorithm because it leverages (1) semidefinite programming, (2) a trust region framework, and (3) explicit constraint handling. As such, the CUATRO framework can be used flexibly.

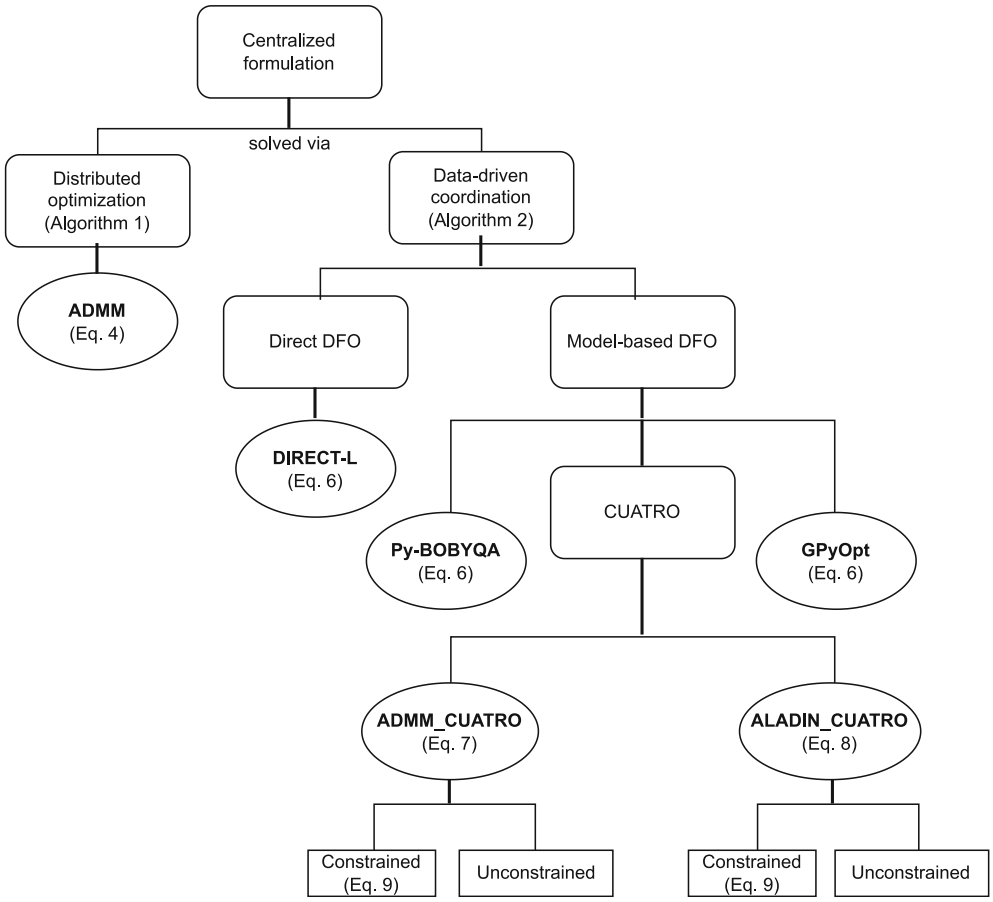

FIGURE 2 Classification of considered algorithms with associated problem formulations in parentheses: We can decide to solve our equivalent centralized formulation either via distributed optimization, namely ADMM, or via our proposed data-driven coordination framework. Within our framework, we can choose any derivative-free optimization (DFO) algorithm. We choose DIRECT-L as a direct DFO algorithm. Among model-based DFO algorithms, we consider Py-BOBYQA and CUATRO as quadratic trust region frameworks and GPyOpt as Bayesian Optimization. We also distinguish between ADMM_CUATRO and ALADIN_CUATRO in the way quadratic surrogates are formulated. Each CUATRO version also has the choice of explicit constraint satisfaction

Explicit constraint handling

When CUATRO is used with explicit constraint handling, the local copies of the shared variables $\mathbf{z}_i$ in the objective evaluation and constraints from (6) are replaced by the exact shared variables $\mathbf{z}$.

$$F_i(\mathbf{z}) = \min_{\mathbf{x}_{i,p}} f_i(\mathbf{x}_{i,p}, \mathbf{z}) \quad \text{s.t.} \quad \mathbf{g}_i(\mathbf{x}_{i,p}, \mathbf{z}) \leq \mathbf{0} \tag{9}$$

where $F_i(\mathbf{z})$ is a tuple consisting of the objective and binary feasibility evaluation: $F_i : \mathbb{R}^{n_z} \to \mathbb{R} \times \{0, 1\}$. 1 denotes if the evaluation for $\mathbf{z}$ is feasible. This makes the subproblem less robust to local constraints, but by returning solver status as feasibility evaluations on top of objective evaluations, we can map the feasibility space in CUATRO by quadratic discrimination and hence concentrate the search on the expected feasible space. Quadratic discrimination—closely related to linear discrimination but using quadratic discriminants—tries to find the "hyperellipsoid" in the input space that best separates samples belonging to different classes.[42] As such, quadratic discrimination is used to classify the binary feasibility evaluations arising from the success or failure of subproblems. ADMM_CUATRO and ALADIN_CUATRO are used with explicit constraint handling if a feasible starting point can be found.

In summary, ADMM_CUATRO and ALADIN_CUATRO leverage Formulations (7) and (8) in their surrogate construction respectively. In both cases, the subproblems are formulated using Formulation (6) unless a feasible initial guess is known and (9) is used. While we benchmark data-driven ADMM and ALADIN against conventional ADMM, we also benchmark them against other DFO algorithms for the upper-level coordination instead of CUATRO. All other DFO algorithms (Py-BOBYQA, GPyOpt, DIRECT-L) are presented in the next section and use the general data-driven coordination framework (6) without explicit constraint handling. The choice between (7) and (8) is not relevant for DIRECT-L as it is not a model-based method, and Py-BOBYQA and GPyOpt can only be used in the single-surrogate data-driven framework (7).

2.4.3 | Py-BOBYQA

On top of CUATRO, we include another trust-region based method. van de Berg et al.[32] show that Py-BOBYQA[43,44] can be competitive with state-of-the-art DFO Python implementations especially in higher-dimensional deterministic case studies. Py-BOBYQA is a Python implementation of Powell's BOBYQA. It iteratively constructs a linear-quadratic regression-interpolation model for the objective, and determines the next step by minimizing said model within a trust-region framework. The user can manipulate how many evaluations are used for each surrogate, determining if the surrogates used resemble more linear or quadratic surrogates. We use Py-BOBYQA with its standard options but enable the multiple restarts heuristic to avoid getting stuck in local minima.

2.4.4 | GPyOpt

Apart from exploitative, trust-region model-based DFO solvers, we also include a Bayesian optimization (BO) implementation as a different type of model-based DFO. BO is generally regarded as the go-to framework for black-box optimization within chemical engineering[45–50] due to its data efficiency and ability to navigate the exploration-exploitation trade-off. As such, BO manages to make significant progress in few evaluations. However, it is known to scale poorly with the number of dimensions and evaluation budget.[32] Informed by Cartis et al.,[44] we are using GPyOpt as our implementation as we prioritize convergence within the low-accuracy regime given our tight budget. GPyOpt[51] is a Python open-source library of BO and builds on GPy, a Python framework for Gaussian process modeling. We use GPyOpt with its default hyperparameters. The interested reader is referred to[52] for more information on Gaussian Processes and Bayesian Optimization.

2.4.5 | DIRECT-L

Finally, we also include a "direct" (model-free) DFO method. Informed by van de Berg et al.,[32] we choose DIRECT-L as a competitive direct solver which displays consistency in convergence and a good exploitation-exploration trade-off. This work's randomized DIRECT-L implementation is taken from the NLopt nonlinear optimization package library.[53] This implementation is based on the 1993 DIviding RECTangles algorithm for global optimization, originally written in FORTRAN.[54] DIRECT is a Lipschitzian, deterministic search algorithm, based on systematic partitioning of the search space into smaller hyperrectangles. DIRECT-L can in theory find the global optimum to DFO problems when given infinite function evaluations.[55] In our examples, DIRECT-L often finds global optimal solutions with a limited number of function evaluations, even if this cannot be guaranteed. Thus, DIRECT-L is also supposed to converge to the global optimum in our suggested framework given enough evaluations. Gablonsky and Kelley[56] then made the algorithm biased toward local search for problems that only have a few local minima. Johnson's NLopt's implementation uses a randomized version of the locally biased DIRECT, which involves randomness in deciding on the dimension to partition along next when function evaluations are close.

2.5 | Algorithms and software implementation

We use Pyomo[57,58] as Python-based optimization software with the numerical solvers Ipopt[59] or Gurobi[60] to optimize the continuous or mixed-integer lower-level subproblems given by (4), (6), or (9). Information from these problem instances are then extracted to be used in the upper-level distributed optimization or DFO. We use readily available Python packages for GPyOpt, Py-BOBYQA, and DIRECT-L, and an in-house Python implementation of ADMM and CUATRO. The generalized framework for our proposed framework and its comparison to ADMM is found in Figure 1, while Figure 2 illustrates how the DFO methods fit into our framework. The code for the algorithms and benchmarking is available under https://github.com/OptiMaL-PSE-Lab/Data-driven-coordination.

2.6 | Game-theoretical and other considerations

Coordination problems are interdisciplinary in nature, and are rooted in a rich body of literature within the field of game theory.[61,62] While we are less interested in the game-theoretical underpinnings of these problems, we need to state some assumptions that justify our proposed method and investigated case studies. First, ADMM and our proposed "data-driven coordination" techniques involve an upper-level, centralized "coordination" step. This presumes the existence of a coordinator agent or software that is acting in good faith, which should be a reasonable assumption in EWO. We are also assuming that all agents are honest-but-curious, that is, that no agent is trying to trick the coordinator or launch any adversarial "attacks," which is the scope of a whole subfield of literature.[63]

Finally, we want to acknowledge that coordination within business settings is subject to many different kinds of other considerations: While we investigate algorithms that share as little information as possible, the coordinator-agent and indirectly agent–agent exchange requires an involved legal framework and software infrastrucutre.[64] While ADMM and our proposed data-driven coordination algorithms in principle allow for privacy-preservation, this would in practice require a thorough investigation into differential privacy and cryptography schemes. The interested reader is referred to Rodríguez-Barroso et al.[65] and[66] for a thorough discussion.

3 | CASE STUDIES

Data-driven coordination is expected to shine in applications that are low-dimensional and nonconvex. As such, we start with a motivating example before presenting three EWO-specific examples. We use the motivating example in Section 3.2 to provide an ideal use case for data-driven coordination and illustrate how decomposition might work in practice for distributed optimization and data-driven coordination. We then benchmark all solvers on this motivating example and investigate the effect of starting points and the penalty parameter ρ on convergence before comparing the algorithm solution times and overhead of all methods. In the collaborative model training in Section 3.3, we analyze the effect of increasing the number of agents and shared variables. In the facility location problem in Section 3.4, we show the effect of different organizational considerations and increasing subproblem size on the convergence. Finally, in the multiagent coordination in Section 3.5, we analyze the effect of (non)convexity and solution topology on the convergence. In Section 4, we then summarize the case-study specific results.

3.1 | Convergence plots

For each application, we lay out the case study with any relevant background before we present the numerical experiments and their results. For each case study configuration, we show best function evaluation vs. number of function evaluation and convergence to the centralized solution optimum vs. number of function evaluation plots

for all algorithms. We use number of function or subproblem evaluations interchangeably with iterations since in both frameworks, each agent solves their subproblem as an optimization instance at each iteration. This way we also normalize against the computational time required for the subproblem solutions. This type of analysis hinges on the assumption that convergence relies only on the number of shared variables and the topology of the best function evaluation to shared variable mapping rather than the size of the subproblems. This will be discussed in greater detail in Section 3.4.6.

Since the underlying subproblems are deterministic, we only need to include a single realization of the deterministic methods ADMM and Py-BOBYQA. While CUATRO randomly samples function evaluations within the trust region, we also only include a single realization of both CUATRO versions at the default seed. For DIRECT-L and GPyOpt however, we include five realizations each on all case studies. We plot their median evaluation with their min–max range shaded. While the best function evaluation plots illustrate low-accuracy convergence (especially relevant for a tight function evaluation budget), convergence plots are needed to compare high-accuracy convergence.

3.2 | Motivating example

We first consider the following synthetic toy problem:

$$\begin{aligned} \min_{x_1,x_2,x_3} \;& (x_1-7)^2+(x_1x_3-3)^2+(x_2+2)^2+(x_2x_3-2)^2 \\ \text{s.t.} \;& x_1 \geq 0,\; x_1+x_3=5 \\ & -10 \leq x_1,x_2,x_3 \leq 10 \end{aligned} \tag{10}$$

We can see that after fixing x_3, the problem becomes trivially separable into two subproblems. This means that this problem can be reformulated into a one-dimensional DFO problem. As such, we introduce z to take the place of x_3, and introduce local copies of z, namely z^I and z^{II}. Then, we penalize the deviation between z and its local copy using a proximal term in the objective:

$$\begin{aligned} (1)\;\; F_1(z) = \; & \min_{x_1,z^I} \; (x_1-7)^2+(x_1z^I-3)^2+\frac{\rho}{2}(z^I-z)^2 \\ & \text{s.t.} \;\; x_1 \geq 0,\; x_1+z^I=5,\;\; -10 \leq x_1,z^I \leq 10 \\ (2)\;\; F_2(z) = \; & \min_{x_2,z^{II}} \; (x_2+2)^2+(x_2z_2^{II}-2)^2+\frac{\rho}{2}(z^{II}-z)^2 \\ & \text{s.t.} \; -10 \leq x_2,z^{II} \leq 10 \end{aligned} \tag{11}$$

Py-BOBYQA, DIRECT-L, GPyOpt aim to find $\mathbf{z}$ that optimizes $F_1(z)+F_2(z)$. ADMM uses the same subproblems with the exception that the proximal term includes the addition of u^I and u^{II} following (4). For a trivial problem like this, we can find a feasible starting point in the upper-level variables for Problem (10). We use $z=4.5$ as starting point, for which we can find x_1 and x_2 in the subproblems that satisfy all of the constraints. As such, we use ADMM_CUATRO and ALADIN_CUATRO in its constrained form, meaning that we omit z_I and z^{II} as decision variables, omit the proximal term in the subproblem objective, replace z^I and z^{II} with z in the objective and constraints, and return the solver status as a binary feasibility evaluation on top of the objective as described in (9):

$$\begin{aligned} (1)\;\; F_1(z) = \; & \min_{x_1} \; (x_1-7)^2+(x_1z-3)^2 \\ & \text{s.t.} \;\; x_1 \geq 0,\; x_1+z=5,\;\; -10 \leq x_1,z \leq 10 \\ (2)\;\; F_2(z) = \; & \min_{x_2} \; (x_2+2)^2+(x_2z-2)^2 \\ & \text{s.t.} \; -10 \leq x_2,z \leq 10 \end{aligned} \tag{12}$$

Figure 3, which plots the upper-level objective evaluation of the bilevel formulation (3) as a function of the shared variable z, shows an inflection point around $z=3.75$ which can hinder the convergence of ADMM and hence call for our proposed methods.

3.2.1 | Base case

The motivating example encompasses all of the properties that call for data-driven coordination. The problem uses a penalty parameter ρ of 1000, is one-dimensional with a starting point at $z=4.5$, and an inflection point around $z=3.5$, which might hinder convergence of purely exploitative methods. Figure 4C shows the solution space convergence of CUATRO and ADMM. While ADMM fails to pass the inflection point, all data-driven methods apart from ADMM converge to a near-optimal solution. The best function evaluation plot (Figure 4A) shows that the DFO variants converge to a low-accuracy solution in the following order from first to last: DIRECT-L, GPyOpt (Bayesian Optimization), Py-BOBYQA, ALADIN_CUATRO, and ADMM_CUATRO. In this one-dimensional case study, initial exploration in DIRECT-L and GPyOpt encourages escaping the saddle point as quickly as possible. Figure 4B then shows that both CUATRO versions achieve a convergence of around 10^{-8} and 10^{-10} for the ADMM and ALADIN versions respectively, while the other DFO variants only achieve a median convergence up to 10^{-3} or 10^{-5}. This is due to the small trust region radius of the two CUATRO versions in later iterations favoring fine-tuning.

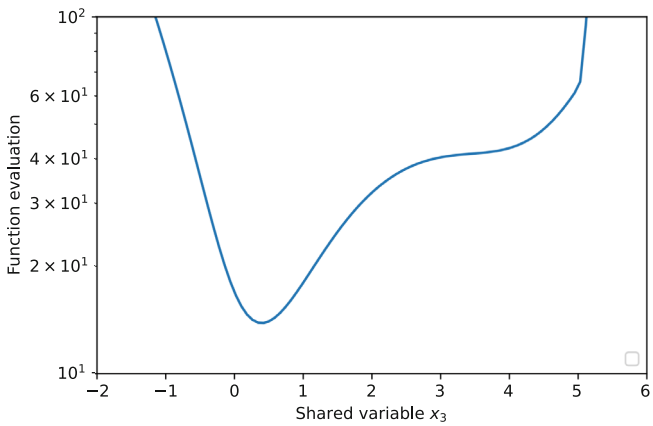

FIGURE 3 Upper-level objective of the motivating example as a function of the shared variable

3.2.2 | Effect of the starting point

In general, the relative performance of distributed optimization and data-driven coordination depends on the initial guess. We repeated the analysis on the same motivating example using the same penalty parameter, but starting from an initial guess of $z = -5.0$ instead.

In Figures 4A and 5A, we see that the low-accuracy convergences of the DFO variants are very similar between the two starting points. When starting from $z = -5.0$, the more exploitative solvers ADMM_CUATRO, ALADIN_CUATRO, and Py-BOBYQA converge in around 10 fewer evaluations. This is expected since the solution space with the initial guess seems to be better-behaved to the left of the optimum (Figure 3). ADMM however displays better performance and manages to converge to a low-accuracy solution in 50 evaluations, since it does not get stuck at an inflection point. While there might be a specific penalty parameter tuning where ADMM could outperform most DFO methods, the DFO methods can be said to more efficiently explore the solution space than ADMM when the number of shared variables is this small.

3.2.3 | Effect of the penalty parameter

A major drawback of ADMM is its sensitivity to the penalty parameter ρ. In theory, convergence is guaranteed for convex problems no matter the value of the penalty parameter. In practice however, if the penalty parameter ρ is too high, the local copies of the shared variables $\mathbf{z}_i$ are restricted in how much they can move away from the suggested set $\mathbf{z}$ at each iteration leading to slow convergence. If the penalty parameter ρ is too low and the local copies are allowed to move too much, then the convergence might display oscillating and—in nonconvex problems—even divergent behavior. These phenomena can be observed in Figure 5B,C for $\rho = 100{,}000$ and $\rho = 10$ respectively compared to the base penalty parameter of $\rho = 1{,}000$. The difficulty of tuning ρ is exacerbated by the nonconvex nature of the motivating example. ρ only manages to escape the local

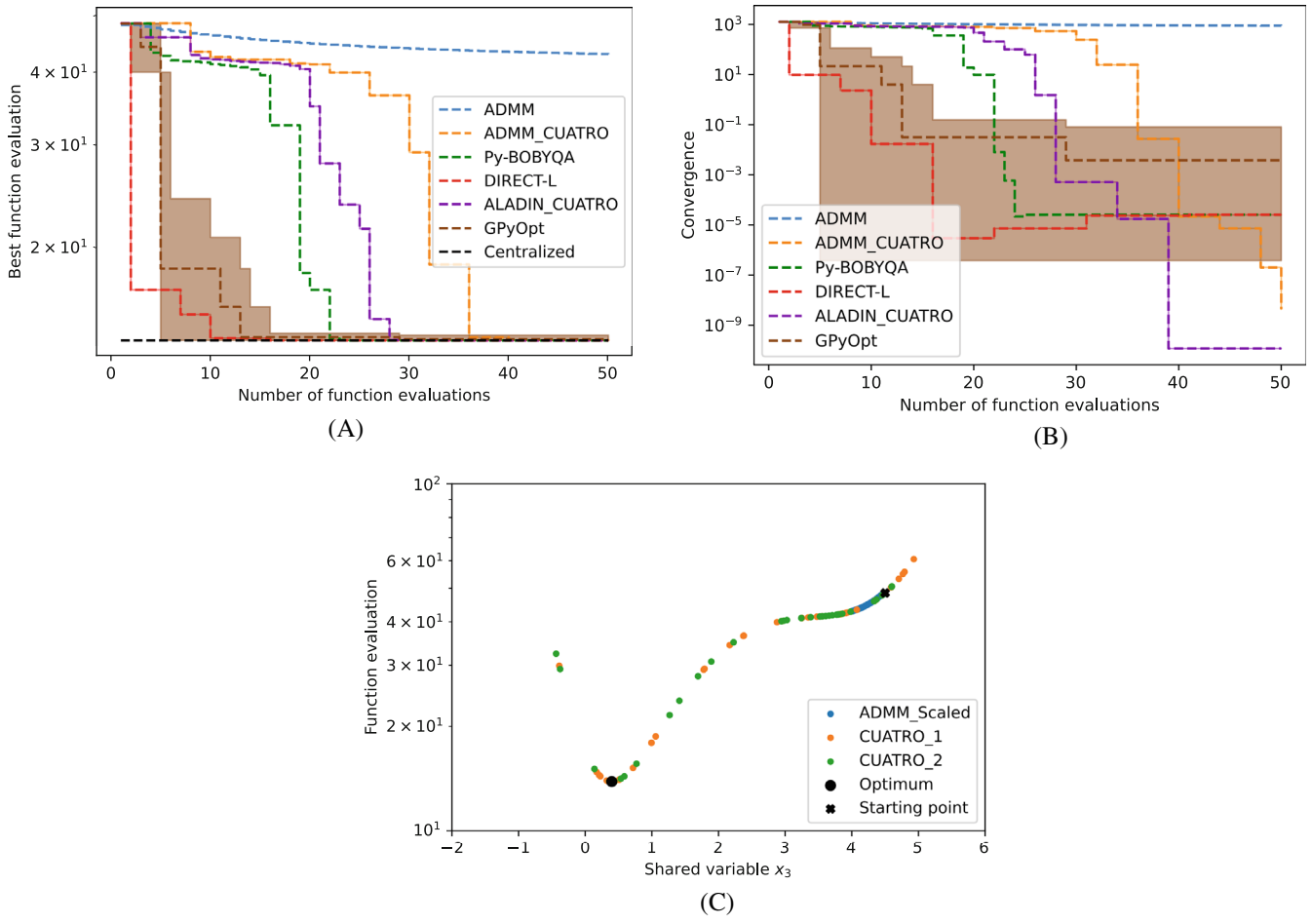

FIGURE 4 Convergence plots for the motivating example starting at $z = 4.5$ and $\rho = 1000$. For DIRECT-L and GPyOpt, the median best evaluation with shaded min–max range is given over 5 runs. (A) Best function evaluation versus number of function evaluations for all methods. (B) Convergence versus number of function evaluations for all methods. (C) Evaluation versus shared variable solution space for ADMM, ADMM_CUATRO, and ALADIN_CUATRO

minimum of the saddle point if the penalty parameter is low. However, this leads to "oscillation" around the solution. If ρ is too high, ADMM might display too slow convergence even to the *local* minimum. Given this difficulty, in the following case studies, we do not perform extensive hyperparameter tuning on ρ. We start from a small ρ and increase it until an acceptable solution is found.

The relative performance of the DFO methods generally displays less sensitivity to ρ. When comparing Figures 4A and 5B, we see that the convergence of DFO algorithms is only slightly slowed down on the motivating example. There is still a trade-off in the choice of ρ: High penalty parameters might lead to ill-conditioned solution spaces when some suggested shared variables are excessively penalized and lead to different orders of magnitudes in the evaluations; Low penalty parameters have another potential problem in both ADMM and DFO solvers. When ρ is small, the penalty occurred by having the local copies $\mathbf{z}_i$ deviate from their proposed set $\mathbf{z}$ could be penalized less than the gain in objective value achieved at the optimal $\mathbf{z}_i$ rather than $\mathbf{z}$. This means that we can find slightly better evaluations than in the centralized formulation. This could manifest as sudden increases in the convergence plots as is the case for DIRECT-L in Figure 4B after around 20 evaluations. In practice, this deviation between the local copies and proposed set of shared variables should be within accepted numerical tolerances for industrially relevant applications.

3.2.4 | A note on computational time

The overhead of ADMM and DFO solvers in coordinating the solution of the subproblems becomes negligible as the size of the subproblems increases. Even on very cheap problems, the algorithm overhead of most methods is insignificant compared to the total subproblem solution time: In the motivating example, we have two subproblems that require on average 0.3583 s (in sequence) to be solved at each iteration or "function evaluation." Since we are using an evaluation budget of 50 in the motivating example, the average total subproblem solution time is around 17.92 s. The total algorithm solution times and overhead times of all methods are given in seconds on the motivating

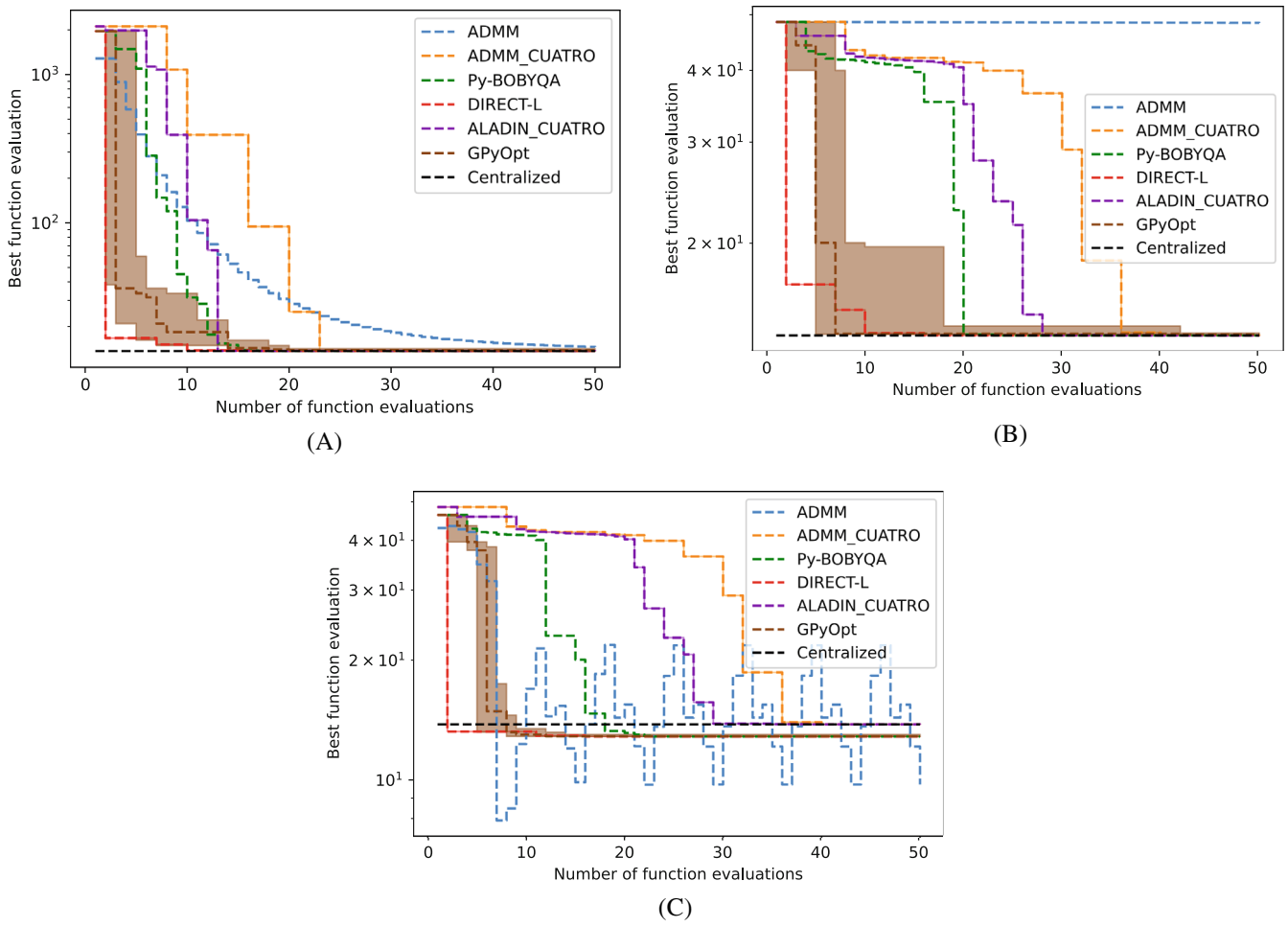

FIGURE 5 Best function evaluation vs. number of function evaluations convergence plots for different variations of the motivating example. For DIRECT-L and GPyOpt, the median best evaluation with shaded min–max range is given over 5 runs. (A) Motivating example starting at $z = -5.0$ and $\rho = 1000$. (B) Motivating example starting at $z = 4.5$ and $\rho = 100{,}000$. (C) Motivating example starting at $z = 4.5$ and $\rho = 10$

problem as follows: DIRECT-L (17.95/0.03), ADMM (18.37/0.45), Py-BOBYQA (18.76/0.84), ADMM_CUATRO (20.95/3.04), ALADIN_-CUATRO (24.95/7.04), GPyOpt (50.04/32.13).

DIRECT-L and ADMM have the best solution times since they do not require optimization in their coordination step to propose the new iteration of shared variables as opposed to the model-based DFO methods. Bayesian optimization solvers are also known to present significant overhead when the evaluation budget is high.[32] As such, in small case studies, the overhead of some methods, especially GPyOpt, might become the bottleneck. However, it is safe to assume that the algorithm overhead becomes negligible in most practical subproblems where total subproblem solution times exceed tens of seconds.

3.3 | Collaborative model training

3.3.1 | Federated learning

The first case study is motivated by Federated Learning (FL).[67] FL is a subfield within Machine Learning (ML), popularized by Google, where multiple clients collaborate under the supervision of a centralized coordinator to train a model while respecting privacy considerations. As such, the aim could be to train a text prediction ML model on decentralized edge devices' (i.e., phones) data while preserving user privacy. For deep neural networks, this usually involves an iteration over the following steps as described in the FedAVG and FedSGD[68] algorithms: The model is broadcast to a selection of training agents. The agents perform a model parameter update on local data based on a stochastic gradient descent step obtained by backpropagation. These model updates are then averaged among all participating agents, potentially preceded by an encryption or differential privacy step. The interested reader is referred to Kairouz et al.[64] for an overview of typical FL challenges.

3.3.2 | Cross-silo "learning"

We are more interested in a "cross-silo"[64] rather than "cross-device" setting, where the number of participating agents is fewer but the primary bottleneck resides in the model update computation, rather than in communication. Additionally, first-order methods such as stochastic gradient descent may not always be applicable if models cannot be (cheaply) differentiated for gradient information (i.e., if the model to be trained is constrained or dynamic). As such, we investigate a generalized coordination scheme for collaborative model training using distributed optimization or data-driven coordination. Similar to ADMM, the conventional FL scheme also involves an averaging step of the model parameters as "shared variables." But as opposed to FedAVG,[68] the subproblem local variable update cannot be obtained under closed form. Instead, we fall back on the more general optimization formulation used in (4).

3.3.3 | Case study

Our considered case study is based on[69,70] and addresses collaborative linear regression with a nonconvex truncated loss term augmented by a 1-norm regularization term. The centralized problem is trivially separable such that:

$$F_i(\mathbf{z}) = \min_{\mathbf{z}_i} \frac{\zeta}{2M_i}\sum_{j=1}^{M_i}\left(\log\left(1+\frac{\left(y_{i,j}-\mathbf{z}_i^\top \mathbf{x}_{i,j}\right)^2}{\zeta}\right)\right)+\xi_i\|\mathbf{z}_i\|_1+\frac{\rho}{2}\|\mathbf{z}_i-\mathbf{z}\|_2^2 \tag{13}$$

The truncated loss term is used to make the regression more robust against outliers, while the regularization term penalizes nonsparsity in the regression coefficients. In the linear regression, $\mathbf{x}_{i,j}\in\mathbb{R}^d$ denotes the jth sample's predictors of the ith agent, and are normally distributed. $\mathbf{z}$ denotes the regression coefficients. $y_{i,j}\in\mathbb{R}$ denotes the jth observed data sample of the ith agent, and is synthesized according to $y_j=\mathbf{z}^{*\top}\mathbf{x}_{i,j}+v_{i,j}$ where $v_{i,j}$ is random Gaussian noise with standard deviation spanning a tenth of the number of dimensions. $\mathbf{z}^*$ denotes the ground truth model coefficients, sampled uniformly from $[-1, 1]^d$, where d denotes the dimensionality of the problem, namely the number of model coefficients. We use 3000 data samples in total, such that each of the N agents has $M_i=3{,}000/N$ data samples. ζ and ξ_i, which control the level of truncation and regularization are set to 3 and 0.01 respectively.

3.3.4 | Experiments

The subproblems do not contain any local constraints. The objectives in (13) are again tailored toward the implementation of ADMM according to (4). CUATRO is used in its standard, nonconstrained form because the problem itself is not constrained. Starting from an initial solution of $\mathbf{z}=[0, \ldots, 0]^\top$, we investigate the following configurations of the case studies: We first explore how the comparison of the different algorithms changes with increasing dimensionality ($d=2$,10,50) at a fixed number of agents ($N=2$). The second set of experiments explores what happens when the number of agents is increased ($N=2$,4,8) when the dimensionality of the prediction coefficients is fixed ($d=6$). All configurations use a penalty parameter ρ of 10.

3.3.5 | Effect of the number of shared variables

Much of the algorithm convergence discussion in this section follows that of the motivating example because both problems present a nonconvex objective. The DFO variants perform particularly well on the lower two-dimensional case study (Figure 6A), taking up to 20 evaluations to converge compared to the 100 of ADMM. The convergence plot (Figure 6D) shows that Py-BOBYQA and both CUATRO variants converge to a high degree of accuracy in 20 evaluations, which takes DIRECT-L around 40 evaluations to

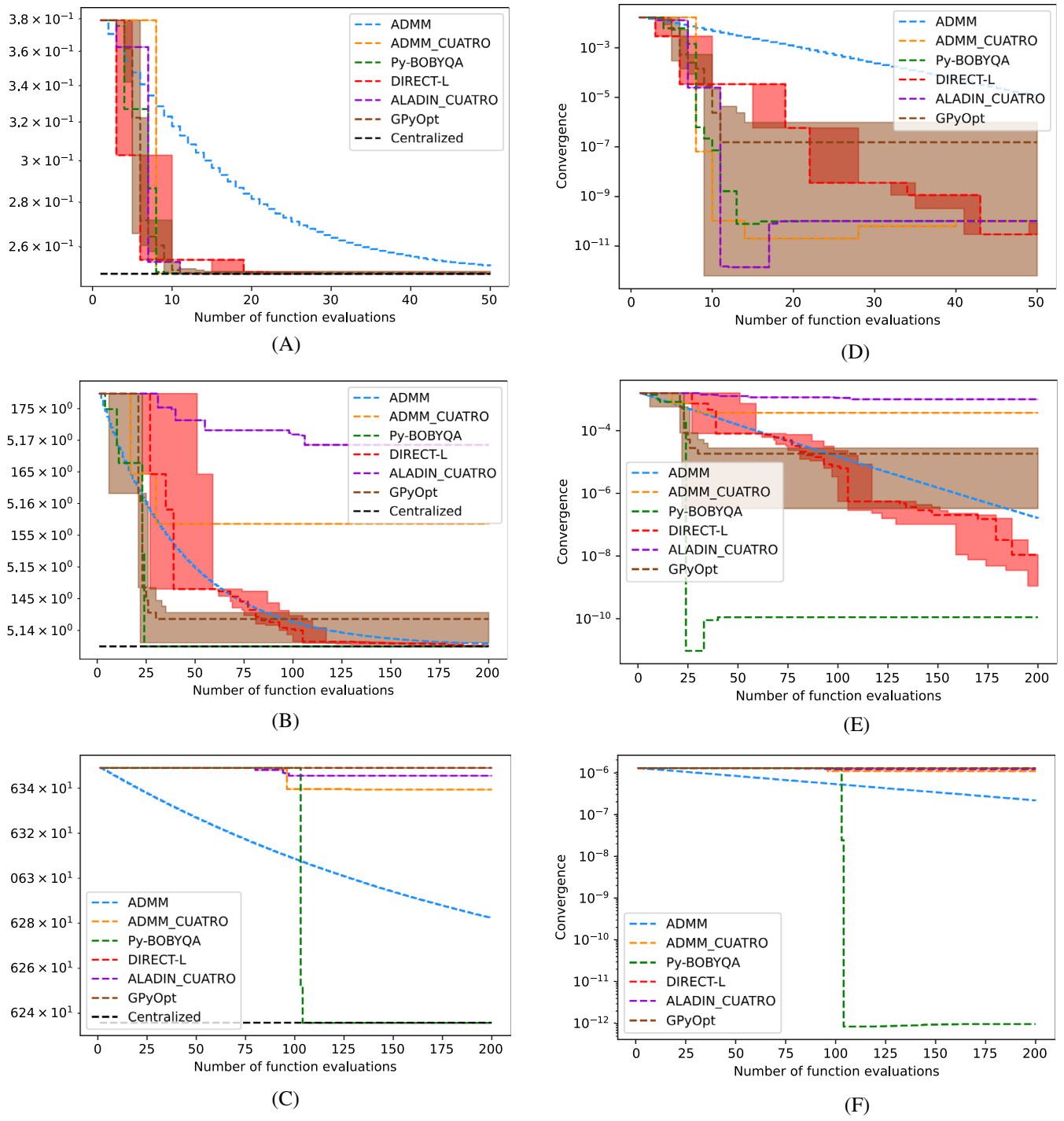

FIGURE 6 Effect of the number of shared variables on the truncated regression case study. For DIRECT-L and GPyOpt, the median best evaluation with shaded min–max range is given over 5 runs. (A) Best function evaluation versus number of function evaluations using 2 agents and 2 shared variables. (B) Best function evaluation versus number of function evaluations using 2 agents and 10 shared variables. (C) Best function evaluation versus number of function evaluations using 2 agents and 50 shared variables. (D) Convergence versus number of function evaluations using 2 agents and 2 shared variables. (E) Convergence versus number of function evaluations using 2 agents and 10 shared variables. (F) Convergence versus number of function evaluations using 2 agents and 50 shared variables

reach. ADMM and GPyOpt only reach a (median) convergence of 10^{-5} and 10^{-7} respectively compared to 10^{-10} of the other three methods. Like in the motivating example, GPyOpt displays significant variance in its final convergence.

Figure 6B shows that when the dimensionality is increased to 10, the CUATRO variants lose their competitiveness with ADMM. Figure 6E illustrates how DIRECT-L displays a similar median convergence speed to ADMM. Py-BOBYQA and GPyOpt make substantial

progress in the first 20 evaluations. The best final convergence realization of GPyOpt matches that of ADMM (10^{-7}), while Py-BOBYQA is the best at fine-tuning solution accuracy (10^{-10}).

In the 50-dimensional case, we would expect ADMM to significantly outperform all DFO variants given its competitive advantage as a subgradient method, which becomes increasingly important in higher dimensions. However, in Figure 6C, we see that after around 100 evaluations, Py-BOBYQA is the only variant to find the optimum. This makes sense given that the starting point of the case study $\mathbf{z} = [0, \ldots, 0]$ is already quite close to the optimal solution. This gives exploitative methods (ADMM and Py-BOBYQA) the upper hand. Finally, ADMM, despite making consistent progress, is slower at fine-tuning the optimum than Py-BOBYQA.

3.3.6 | Effect of the number of agents

Starting with a dimensionality of six and two coordinating agents, we observe a similar convergence pattern in Figure 7A,D to the 10-dimensional case in the previous section (Figure 6B,E). ADMM, Py-BOBYQA, ALADIN_CUATRO and GPyOpt display a similar relative performance, while ADMM_CUATRO makes consistent progress and matches the convergence found for BO and Py-BOBYQA in 35 and 55 evaluations respectively.

Overall, we observe that with an increasing number of coordinating agents, the optimality gap increases between the final total function evaluation and the centralized optimum. There will always be small numerical differences between $\mathbf{z}_i$ and $\mathbf{z}$, which are penalized in the proximal terms $\frac{\rho}{2}\|\mathbf{z}_i - \mathbf{z}\|_2^2$. With an increasing number of agents, the relative importance of these proximal terms is strengthened, which becomes even more apparent when the optimal objective evaluation is close to the starting point as is the case for these problems.

The relative performance of the DFO algorithms with respect to each other and ADMM does however not seem to change with an increasing number of coordinating agents. ALADIN_CUATRO makes poor progress, DIRECT-L tracks the convergence of ADMM, while Py-BOBYQA and ADMM_CUATRO quickly find the best function evaluations. GPyOpt again makes quick initial progress but displays a lot of variance in best evaluation found. It is interesting to see that for this particular case, the convergence speeds of Py-BOBYQA, ADMM_CUATRO, and GPyOpt tend to remain similar even with an increasing number of agents.

3.4 | Facility location

3.4.1 | Value chains

A key part of EWO is the design and operation of supply chains.[71] In an idealized setting, all stakeholders within a given value chain are willing to collaborate and share model information with a centralized coordinator. In practice however, antitrust and game-theoretical considerations might prevent stakeholders from fully collaborating. There is ample literature about "Stackelberg Leader-Follower games,"[72] where supply chain agents' take the first step' in deciding on the optimal location of their plants subject to other players reacting optimally with respect to their private objective. Yet, there is much value to be captured in moving away from these "Nash equilibria," and approaching a coordinated optimum along the "Pareto front." To this end, a coordinator can optimize a (fairness-guided) game-theoretical operator that scalarizes and trades off the conflicting criteria of competing stakeholders.[73–76]

This is especially relevant for the design of emerging supply chains with distinct characteristics such as biomass[77,78] or (bio)pharmaceutical value chains.[79,80] These "social optima" are often obtained as the result of centralized optimization formulations, which can be decomposed for numerical tractability, or in our case to fit organizational considerations.

3.4.2 | Case study

We consider a continuous facility location problem in two-dimensional continuous space, which belongs to the general class of Capacitated Multifacility Weber Problems. The objective is to find the location, production, and connecting flows of all facilities that minimize a total cost. These "shared" variables are few relative to the number of private variables and parameters, the latter including local cost parameters, technical upper and lower bounds, binary variables, and distances to/from facilities to name but a few. We use the same formulation as Lara et al.[81] with some key differences. In particular, we assume the presence of two suppliers and markets each. We fix the number of facilities to be built to either one or two, which still gives rise to a Generalized Disjunctive Problem (GDP). We also define the distances between agents and facilities using the 1-norm, rather than the 2-norm for computational efficiency. The GDP is finally reformulated into an MINLP using big-M constraints, implemented in Pyomo,[57,58] and solved using Gurobi.[60] The entire formulation can be found in Appendix E.

3.4.3 | Decomposition

In our first experiment, we compare two different decompositions, motivated by two separate business scenarios. In the first scenario, the two types of nodes, suppliers and markets, each consisting of two nodes, are part of the same legal entity and are able to share model information. For each problem, we need to find the following shared variables: The two-dimensional location of the facilities ($2K$ variables) and their production (K variables). For our case, where the number of processing facilities is set to one or two ($K = 1,2$), the problem contains either three or six shared variables and can be decomposed into two subproblems. The exact decomposition can be found in Appendix E.

In our second scenario, we consider all of the four nodes (supplier and customer) to be their own separate legal entity with privacy considerations. As such, we need to decompose the problem into four. Unfortunately, the presence of complicating constraints—linking the

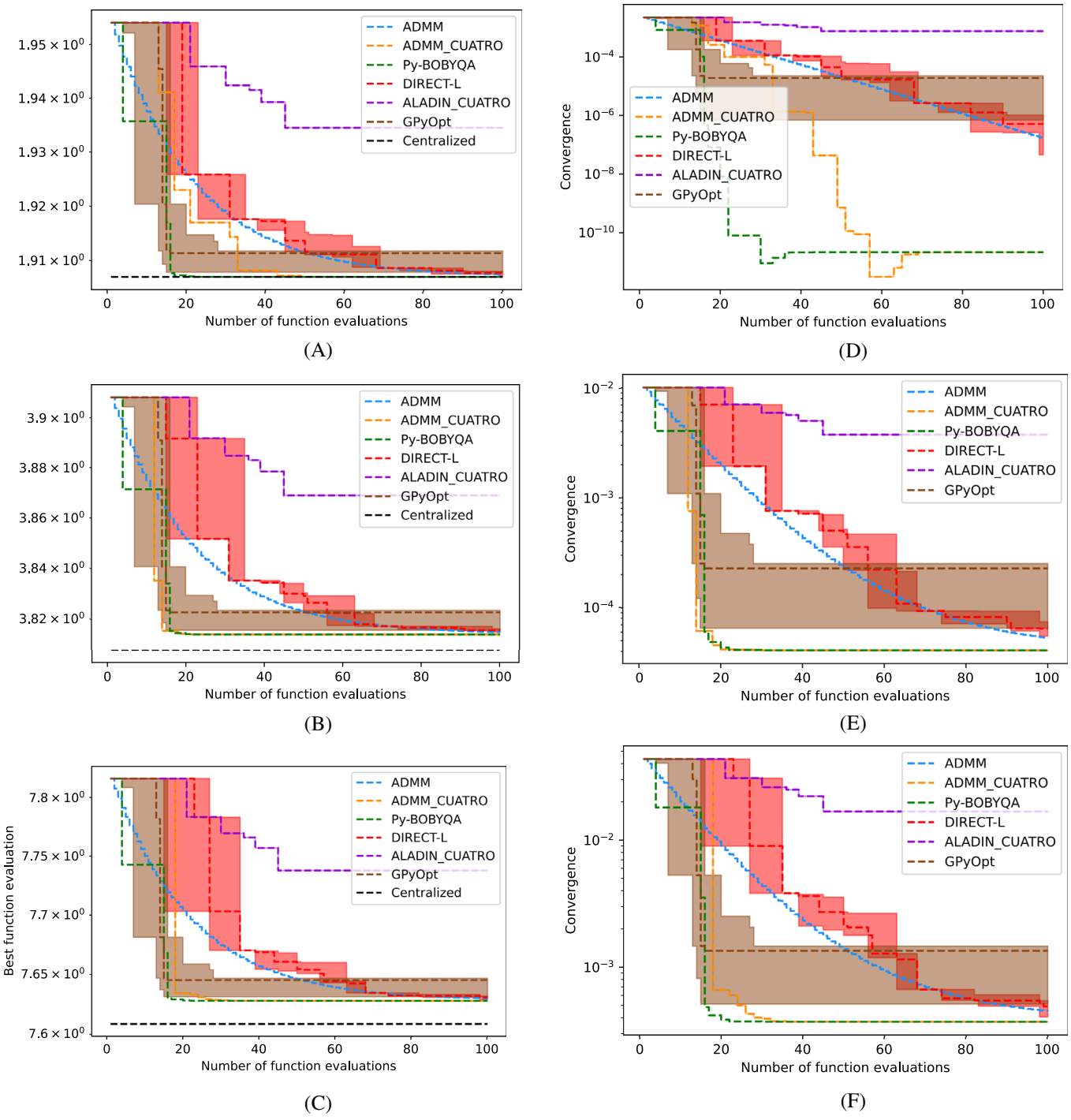

FIGURE 7 Effect of the number of agents on the truncated regression case study. For DIRECT-L and GPyOpt, the median best evaluation with shaded min–max range is given over 5 runs. (A) Best function evaluation versus number of function evaluations using 2 agents and 6 shared variables. (B) Best function evaluation versus number of function evaluations using 4 agents and 6 shared variables. (C) Best function evaluation versus number of function evaluations using 8 agents and 6 shared variables. (D) Convergence versus number of function evaluations using 2 agents and 6 shared variables. (E) Convergence versus number of function evaluations using 4 agents and 6 shared variables. (F) Convergence versus number of function evaluations using 8 agents and 6 shared variables

total amount transported to and from a facility—prevent each agent from independently deciding on the amount that is transported between their node and the facilities. As such, the transport variables $f_{i,k}$ and $f_{k,j}$ become part of the set of shared variables. This would in principle add another $4K$ shared variables. However, these complicating constraints on the transport variables, only depending on the shared variables, can be used to reduce the number of degrees of freedom in the shared variables, such that these problems involve five

or ten shared variables when one or two facilities are built respectively. The exact decompositions can again be found in Appendix E.

In our second experiment, we investigate the effect of increasing subproblem size on the relative algorithm convergence vs. number of subproblem evaluations while keeping the number of shared variables constant. We start from the scenario where we want to build two processing facilities in the presence of two agents, meaning the number of shared variables is set to 6. Then, we increase the number of suppliers and customers from two to five nodes each ($N_i = N_j = 2,3,4,5,6$). This increases the number of private variables and constraints in the subproblems and the CPU time required to solve them. Yet, this should have no major effect on the solution topology given the similar nature of the subproblem optimization structure.

3.4.4 | Two agents: supply and customer nodes belong to the same supply and demand decision-makers

Figure 8A shows that for the two-agent three-dimensional case, the exploitative methods ADMM and Py-BOBYQA display a similar convergence speed and are the only methods to converge quickly to the optimum. The two CUATRO versions converge to the same similar suboptimal point. The median DIRECT-L run manages to find the same optimum as ADMM and Py-BOBYQA after around 90 evaluations. GPyOpt only makes little progress. Figure 8B then shows that in the six-dimensional case, Py-BOBYQA and ADMM again outperform both CUATRO variants and Bayesian Optimization. As expected, the subgradient information of ADMM leads to faster convergence compared to Py-BOBYQA when the dimensionality is increased. Interestingly, DIRECT-L outperforms both exploitative methods, suggesting that the optimum is close to the center of one of the initial partitions used by DIRECT-L, which depends mostly on the user-given box bounds on the shared variables. The convergence vs. number of function evaluations plots are omitted since they do not provide any additional information, as ADMM, Py-BOBYQA, and DIRECT-L converge to around the same accuracy.

3.4.5 | Four agents: each supplier and customer node as a separate decision-maker

Figure 9A,B shows a significantly different relative algorithm performance for the four-agent case to that seen in the two-agent case of the last section. This is partially caused by the inclusion of additional shared variables in the form of facility to agent node transport links that need to be coordinated between all supplier and customer nodes. Additionally, these shared variables introduce ill-behavior in the new solution space, which could be due to the way the shared variables are handled. In the multiagent coordination case study, the number of shared variables is kept the same, but any infeasibilities in the shared variables are implicitly penalized through the proximal term. In this case study, complicating constraints are handled by reducing the degrees of freedom in the shared variables using material balances on the facility nodes. The choice on how best to handle complicating constraints is case study-specific. As a rule of thumb however, reducing the degrees of freedom using constraints favors convergence for data-driven methods, as ADMM struggles to deal with ill-behaved solution spaces. However, the DFO variants are not *guaranteed* to outperform ADMM even in lower dimensions when constraints are handled this way.

In fact, for the lower-dimensional case in Figure 9A, only Py-BOBYQA manages to navigate the solution space better than ADMM and is the only method to converge to the optimum. GPyOpt finds a similar optimum to ADMM, while the other DFO variants display worse performance to ADMM. Figure 9B shows a peculiar convergence pattern for the 10-dimensional case, apart for

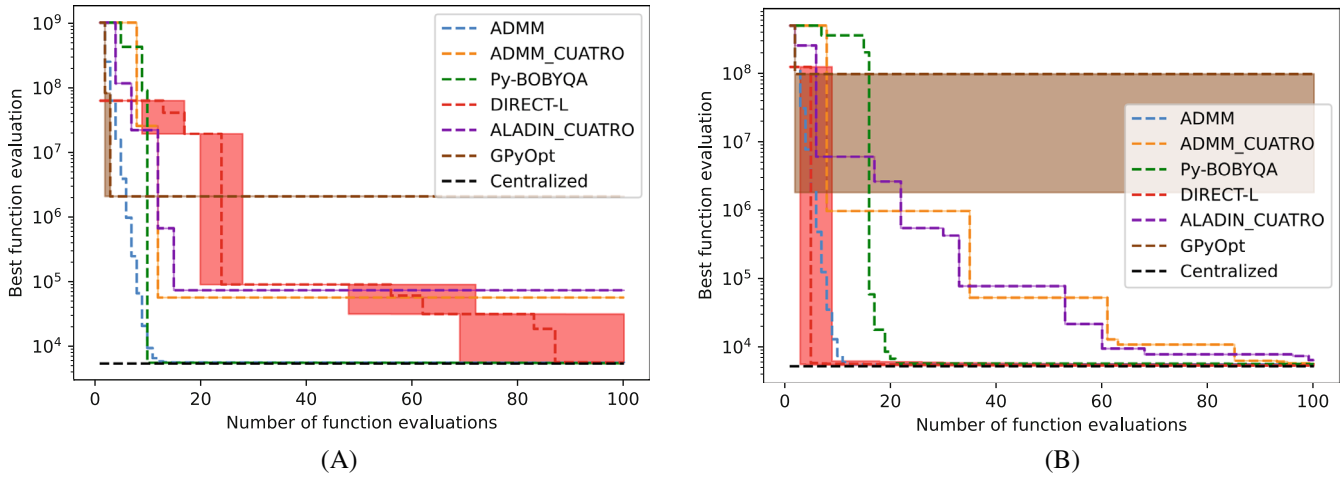

FIGURE 8 Facility location convergence plots with two decision-makers. For DIRECT-L and GPyOpt, the median best evaluation with shaded min–max range is given over 5 runs. (A) Best function evaluation versus number of function evaluations using 2 agents and 3 shared variables. (B) Best function evaluation versus number of function evaluations using 2 agents and 6 shared variables

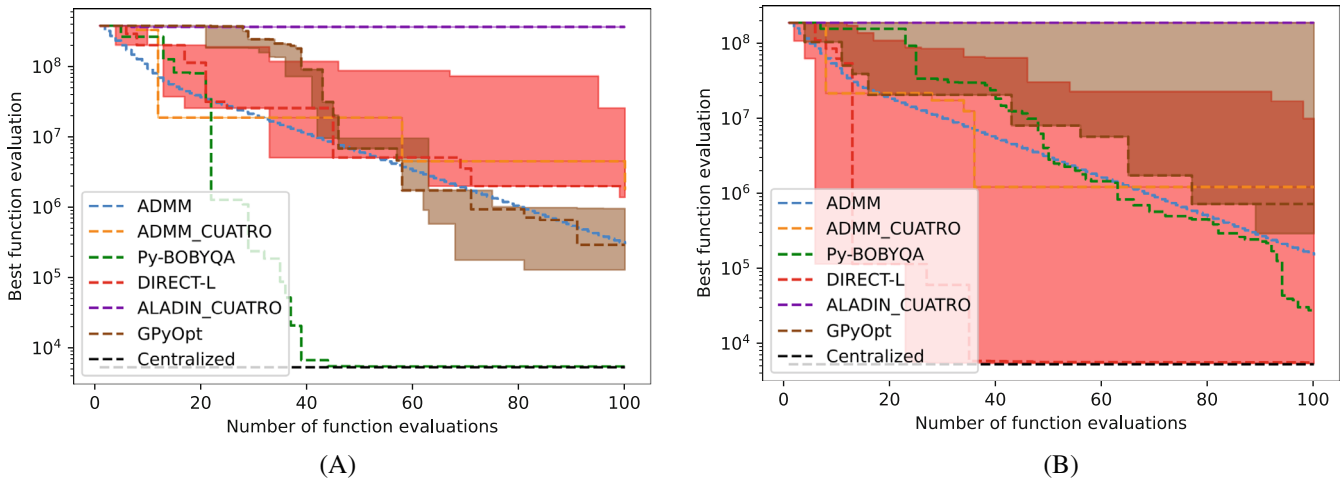

FIGURE 9 Facility location convergence plots with four decision-makers. For DIRECT-L and GPyOpt, the median best evaluation with shaded min–max range is given over 5 runs. (A) Best function evaluation versus number of function evaluations using 4 agents and 5 shared variables. (B) Best function evaluation versus number of function evaluations using 4 agents and 10 shared variables

ALADIN_CUATRO which again makes no progress whatsoever. Py-BOBYQA displays similar convergence patterns to ADMM, and both find a slightly better optimum than ADMM_CUATRO. GPyOpt's final evaluations compete with ADMM. DIRECT-L is the only method to converge to the optimum in its median evaluation but displays significant variability in its convergence. Like in the higher-dimensional two-agent case, the solution space topology and input-bounds give rise to partitions whose center is close to the optimum. Convergence vs. number of function evaluation plots are omitted again as they provide no additional information.

3.4.6 | Effect of the subproblem size

In the previous two- and four-agent configurations, the performance of most DFO solvers would deteriorate relative to that of ADMM as the number of shared variables increases. In Figure 10, we show the effect of increasing size while keeping the number of shared variables constant by increasing the number of supplier and customer nodes rather than the number of processing facilities. We see that the performances of ADMM, Py-BOBYQA, and GPyOpt remain constant for the most part, while the performance of DIRECT-L only slightly decreases in the run of four nodes. The relative performances of ADMM_CUATRO and ALADIN_CUATRO as compared to ADMM deteriorate only in the runs with three and four supplier and customer nodes. This could be explained by the introduction of additional private constraints in the subproblems that can make the solution space more ill-conditioned due to stricter penalization of certain proposed shared variables. Despite the more restricted feasible solution space, the solution topology should not change too much since the subproblems remain mixed-integer quadratic programs where only similar types of variables and constraints are introduced.

Comparing our analyses on increasing the number of shared variables and the subproblem size suggests that DFO algorithms scale better relative to ADMM with an increase in subproblem size rather than number of shared variables. These results also strengthen our argument that the relative performance of ADMM and DFO solvers depends on the evaluation budget, number of shared variables and solution topology rather than subproblem size. As such, our conclusions on the relative performance of ADMM and DFO solvers on smaller case studies extend to larger subproblems as long as the number of shared variables and the solution topologies remain similar.

3.5 | Multiobjective coordination

3.5.1 | Case study

We consider the same synthetic problem as van de Berg et al.[36] where two stakeholders want to find the feedstock composition that optimizes a sum consisting of a cost and environmental impact term. Since both stakeholders are secretive about the intricacies of their proprietary optimization and simulation software, we can either use ADMM or data-driven coordination. In our considered case study, after fixing the feedstock composition variables $\mathbf{z}$, the problem becomes trivially decomposable. Agent A optimizes an economic blending problem, while Agent B optimizes the output of an environmental input simulation.

$$F_A(\mathbf{z}) = \min_{\mathbf{z}^I \in \mathbb{R}^{n_z}, \mathbf{y} \in \{0,1\}^{n_z}} \sum_i^{n_z} \left(z_i^I C_i + \frac{\rho}{2} \left(z_i^I - z_i \right)^2 \right) \tag{14a}$$

$$\text{s.t.} \quad \sum_i^{n_z} z_i^I = 1, \quad \mathbf{l}_{\text{qual}} \le \mathbf{z}^{I\top} A_{\text{qual}} \le \mathbf{u}_{\text{qual}} \tag{14b}$$

$$\mathbf{0} \le \mathbf{z}^I \le \mathbf{y}, \quad \sum_i^{n_z} y_i \le N_{\text{int}} \tag{14c}$$

$$F_B(\mathbf{z}) = \min_{\mathbf{z}^{II} \in \mathbb{R}^{n_z}} \sum_i^{n_z} \left(e_i z_i^{II^{a_i}} + \sum_{j \in J_i} x_i x_j + \frac{\rho}{2} \left(z_i^{II} - z_i \right)^2 \right) \tag{15}$$

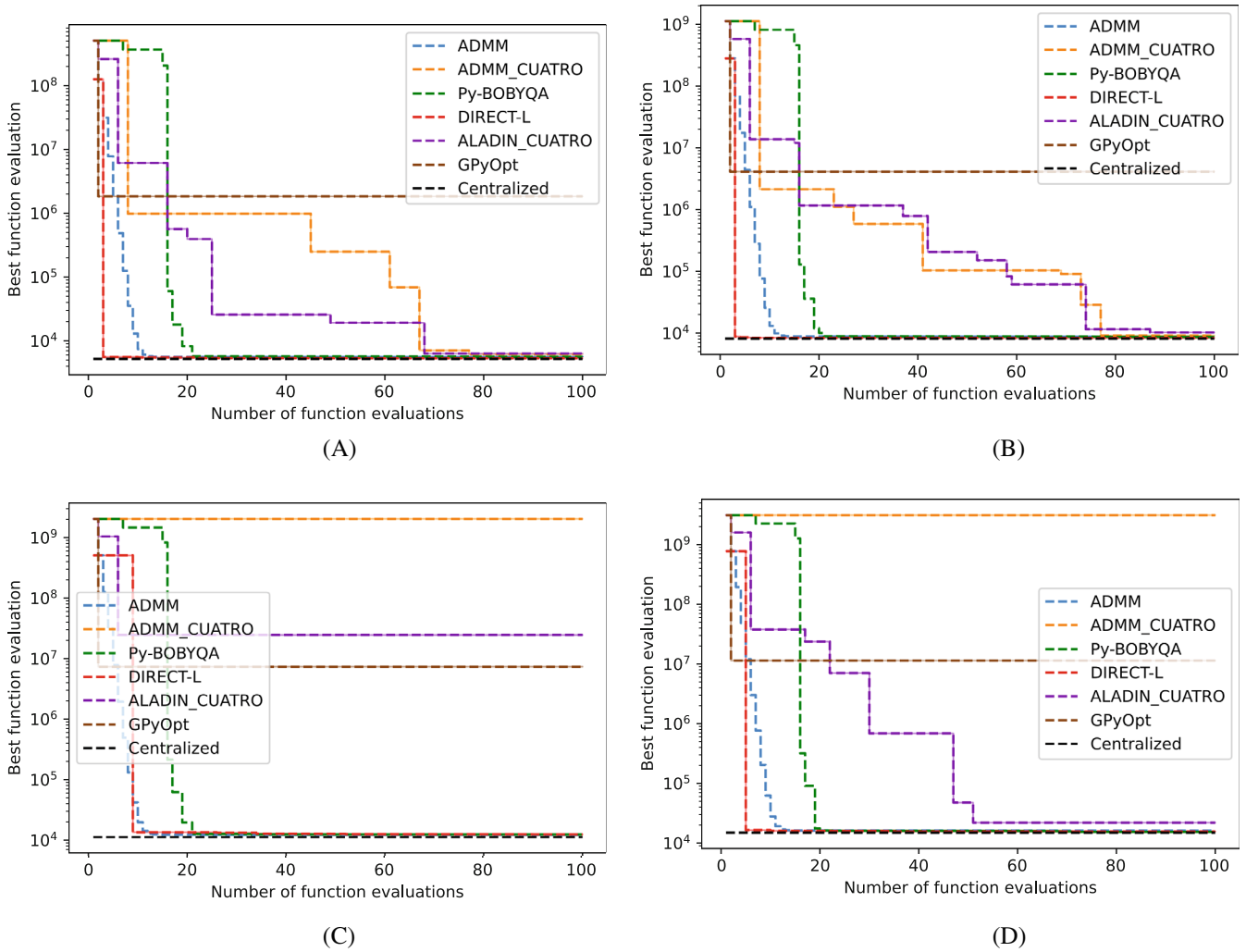

FIGURE 10 Facility location best function evaluation vs. number of function evaluation plots with two decision-makers and six shared variables at with an increasing number of supplier and customer nodes. For DIRECT-L and GPyOpt, the median best evaluation with shaded min–max range is given over 5 runs. (A) 2 nodes per supplier and customer. (B) 3 nodes per supplier and customer. (C) 4 nodes per supplier and customer. (D) 5 nodes per supplier and customer

Essentially, the economic blending problem minimizes the feedstock cost given component costs C_i, as well as upper and lower quality constraints ($\mathbf{u}_{qual}$,$\mathbf{l}_{qual}$) for quality matrix A_{qual}. Each dummy composition variable z_i^I is also subject to its binary variable y_i, which is active if the associated feedstock is non-zero. The number of active composition variables is constrained by N_{int}. As such, Agent A's formulation is a mixed-integer convex quadratic problem. Agent B's objective term is composed of a sum of linear or quadratic terms (each feedstock variable has its own power $a_i \in \{1, 2\}$). Additionally, each feedstock i has its sparse set of bilinear interactions J_i. The cost, quality, and environmental data are adopted from an animal feedstock database.[82]

3.5.2 | Black-box simulations in the subproblems

If the proximal term in the environmental subproblem is strongly penalized with a very high ρ, its optimal solution tends toward the solution corresponding to $\mathbf{z}^{II} = \mathbf{z}$. In this case, since the environmental subproblem does not involve any local constraints, the solution to this problem could theoretically be the result of a black-box simulation rather than optimization problem.

In practice, our data-driven alternatives could readily handle problems where the lower-level is obtained via simulation instead of an optimization, since progress only relies on objective evaluations. ADMM however relies on an update in the local copies of the shared variables. If the lower-level is obtained via a simulation, then $\mathbf{z}^{II}$ is never updated from the suggested $\mathbf{z}$. So at each iteration, $\mathbf{z}$ only approaches $\mathbf{z}^I$ rather than a compromise between $\mathbf{z}^I$ and $\mathbf{z}^{II}$, essentially omitting any environmental considerations. Hence, for ADMM, the subproblems need to be given by optimization. For convergence toward a collaborative optimum, $\mathbf{z}^{II}$ needs to slightly shift away from the suggested $\mathbf{z}$ toward the "selfish" solution of the environmental problem optimized without economic considerations.

As such, in Section 3.5.3, we do not include the case where the subproblem is given by simulation, as this would bias the comparison between ADMM and the data-driven framework in favor of the data-driven coordination.

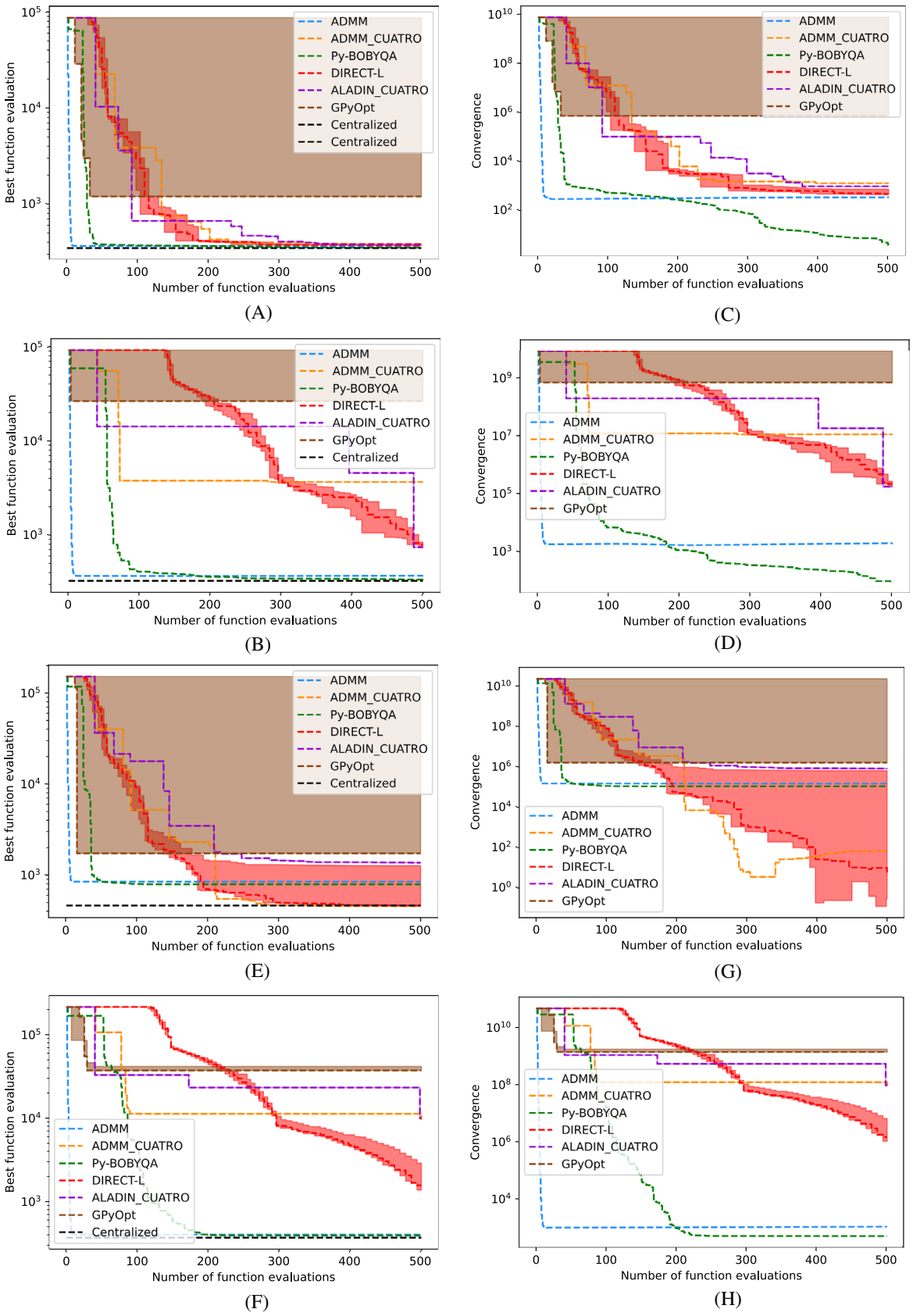

FIGURE 11 Multiagent coordination convergence plots. For DIRECT-L and GPyOpt, the median best evaluation with shaded min–max range is given over 5 runs. (A) Best function evaluation versus number of function evaluations on 10-dimensional convex version. (B) Best function evaluation versus number of function evaluations on 25-dimensional convex version. (C) Convergence versus number of function evaluations on 10-dimensional convex version. (D) Convergence versus number of function evaluations on 25-dimensional convex version. (E) Best function evaluation versus number of function evaluations on 10-dimensional nonconvex version. (F) Best function evaluation versus number of function evaluations on 25-dimensional nonconvex version. (G) Convergence versus number of function evaluations on 10-dimensional nonconvex version. (H) Convergence versus number of function evaluations on 25-dimensional nonconvex version

3.5.3 | Experiments

We perform the mixed-integer, nonconvex coordination problem given by (14) and (15) on an increasing number of shared, feedstock variables ($n_z = 5,10,15,20,25$). Additionally, we perform the same experiment on a nonlinear but convex version of the previous problem. This is obtained by relaxing all mixed-integer constraints in (14c), and by omitting the bilinear terms in the environmental problem (15).

Figure 11 gives the best function evaluation and convergence plots for the 10- and 25-dimensional multiagent coordination problems in their convex and nonconvex variant. All configurations use a penalty parameter ρ of 5000.

Figure 11A,C shows that all methods manage to find at least a low-accuracy optimum in the convex 10-dimensional variant. However, ADMM converges considerably faster than the fastest DFO variant (20 and 200 evaluations for ADMM and Py-BOBYQA to achieve the same accuracy respectively). It makes sense that both ADMM and Py-BOBYQA outperform more explorative methods for the considered configuration given their purely exploitative behavior. This case also highlights the respective strengths of ADMM and Py-BOBYQA. ADMM is in general very fast to converge to a neighborhood of a local optimizer. However, ADMM struggles to fine-tune the optimum. When the evaluation budget allows for it, Py-BOBYQA takes more evaluations to find this neighborhood, but is more efficient at finding a better solution quality. Figure 11C shows that ADMM's final convergence is orders of magnitude worse than that of Py-BOBYQA (10^2 and 10^0 respectively). The discussion of the 25-dimensional convex configuration follows that of the 10-dimensional convex one. The relative performance of the algorithms is very similar, with the exception that ADMM's final convergence still displays a considerable optimality gap (Figure 11B). Py-BOBYQA is the only method to converge to a high-accuracy solution given its exploitative nature and its ability for fine-tuning.

The 10-dimensional nonconvex configuration presents conditions that favor data-driven approaches. The relative performance of the algorithms in Figure 11E is similar to that of its convex counterpart in Figure 11A with two notable exceptions: ADMM and Py-BOBYQA—both purely exploitative methods—converge to a *local* minimizer in 100 evaluations. ADMM is slightly quicker again, but Py-BOBYQA finds a slightly better solution. ADMM_CUATRO and the median run of DIRECT-L, due to their extensive initial exploration manage to escape a local minimizer and converge to a low-accuracy neighborhood of the global optimum. The discussion of the higher-dimensional nonconvex case again follows that of its convex counterpart. ADMM and Py-BOBYQA are the only methods again to converge to at least a near-optimal solution. Py-BOBYQA finds a better solution quality, but takes significantly longer than ADMM. This emphasizes the importance of (sub)gradient information with increasing dimensionality.

4 | GENERAL OBSERVATIONS AND DISCUSSION

In this section, we present general observed trends that summarize the case-study specific observations of Figures 4–11. A high-level comparison between the performance of the data-driven coordination framework and ADMM based on problem considerations is summarized in Table 1.

TABLE 1 Performance comparison between ADMM and data-driven coordination based on mathematical problem and desired solution characteristics

Consideration	ADMM	Data-driven coordination
Number of shared variables **z**	*Scales better* with higher dimensions	Shines in lower dimensions
Convergence speed	*Quick initial progress* but dependent on penalty parameter ρ	Can use exploitative DFO solver to *better fine-tune optimum*
Convergence guarantee	Guaranteed *for convex problems*.	*Depends on DFO solver*, e.g., DIRECT-L for global convergence guarantee
Solution space topology in **z**	Can get stuck at nonconvexities	Can use *explorative DFO* solver to *escape local minima*
Organizational and software	Requires numerical optimization for the subproblem solution	*More flexibility* in the subproblem solution (black-box simulation, heuristic evaluation, etc.)

ADMM. ADMM manages to converge to at least a *local* minimizer if given enough function evaluations. If the proposed starting point is far from the optimum, initial progress with ADMM is generally fast. However, ADMM is found to be ill-suited for fine-tuning near-optimal solutions, which is in line with literature.[19,21]

Data-driven coordination. The performance of all data-driven coordination alternatives improves with respect to ADMM the more ill-behaved the solution space and the lower the dimensionality in the shared variables is. For the EWO case studies, we investigate a lower- and higher-dimensional configuration respectively, where for the lower-dimensional case, there is always at least one DFO variant that outperforms ADMM. Understanding the way these algorithms approach the exploration-exploitation trade-off is key to this observation. The coordination step in ADMM is purely exploitative. It cheaply extracts subgradient information from the subproblems to approach the coordinated optimum as quickly as possible. The relative performance of DFO algorithms against ADMM and explorative vs. exploitative methods is determined by the mathematical properties of the case study.

DFO variants. Highly explorative frameworks like Bayesian Optimization perform well in lower-dimensional applications, where thorough exploration is more likely to be rewarded by faster convergence to the optimum. The exploration of some DFO algorithms can also be useful in escaping local optima. DIRECT-L, as a global optimization algorithm, usually makes slow progress as its function evaluations are used to thoroughly explore all partitions of the solution space, unless the optimum happens to be in the center

of one of the initial partitions. CUATRO, with its decreasing trust region framework, encourages extensive initial exploration and later exploitation. ADMM_CUATRO usually displays superior performance over ALADIN_CUATRO. However, the choice between these two variants is in practice more motivated by organizational considerations concerning the sharing of either agent-level objective or surrogate information.

Py-BOBYQA. The previously discussed DFO algorithms tend to outperform ADMM only in lower-dimensional applications. Py-BOBYQA is the only DFO algorithm that has the potential to be competitive with ADMM in higher-dimensional applications given its similar focus on exploitation. As such, Py-BOBYQA tends to converge to the same local optima as ADMM. While ADMM displays faster convergence to low-accuracy regimes of the solution, Py-BOBYQA can find higher-accuracy solutions at the expense of more function evaluations.

Significance of the penalty parameter ρ. In theory, the value of the penalty parameter ρ should not influence the quality of the solutions found. In fact, the penalty term should approach zero at the optimum, since the local copies of the shared variables $\mathbf{z}_i$ should approach the suggested shared variables $\mathbf{z}$. In practice however, the choice of the penalty parameter ρ influences the accuracy and speed of convergence. If ρ is too weak, more deviation between $\mathbf{z}_i$ and $\mathbf{z}$ is allowed at the theoretical optimum, which can lead to potential infeasibility in the returned solution. Some DFO variants find a better total evaluation than would theoretically be possible from the centralized solution, which explains why some algorithms do not display monotonically decreasing performance in the convergence plots. However, increasing ρ slows down convergence, since at each iteration $\mathbf{z}_i$ is bound closer to $\mathbf{z}$. As such, the conclusions on the relative convergence of the considered methods are influenced by ρ. There is ample literature on how ρ influences the convergence of ADMM.[38] There are multiple heuristics that can speed up ADMM, such as iteratively increasing ρ to allow for more exploration and faster convergence initially while encouraging fine-tuning in later iterations.[39] However, ρ is kept constant across our algorithm benchmarking since our analysis is based on comparing function evaluations. In ADMM, each sample should in principle make consistent progress toward a local minimizer. For the DFO methods however, we have no guarantee that the points we are sampling in later iterations present better function evaluations. By changing ρ between iterations, the same sample would if resampled in a later iteration potentially return a worse evaluation due to a higher ρ. Hence, from the perspective of DFO solvers, changing ρ between iterations would lead to "noise" or "stochasticity" which would hinder their convergence.

5 | CONCLUSION

Our proposed "data-driven" framework is shown to be able to find the same solution as the equivalent centralized formulation for optimization-based coordination problems. Our approach differs from ADMM in that it uses derivative-free optimization (DFO) to find the shared variables that optimize lower-level subproblem evaluations. We consider CUATRO, Py-BOBYQA, DIRECT-L, and GPyOpt as DFO solvers and benchmark them against ADMM as a distributed optimization solver on a motivating example and three case studies with expensive subproblems. We examine the effect that dimensionality and solution topology in the shared variables have on the relative algorithm performance. We also discuss organizational considerations and how they inform the choice of coordinating algorithm: autonomy and flexibility, privacy, software, black-box subproblems, and organizational structure. We show that our approach outperforms ADMM when the number of shared variables between agents is few, and when the shared variable to shared objective evaluations call for exploration rather than exploitation. As opposed to distributed optimization, our method does not need the capacity for numerical optimization at the agent-level, since the subproblems can also be obtained as the result of a black-box objective simulation. We argue that our approach is especially relevant when the decomposition is limited by organizational rather than numerical considerations.

Our work is the first to benchmark several "data-driven" algorithms against distributed optimization on multiple case studies relevant to enterprise-wide optimization. While the relative performance of DFO algorithms is in line with current literature, there are several avenues for future work: A practical implementation of a "data-driven" approach would require a more thorough investigation into privacy (differential privacy, cryptography, and more) and into how much agent-level data could be inferred from optimal subproblem evaluations or variables. ADMM loses its convergence guarantees when the subproblems are ill-behaved. As such, our data-driven optimization approach might have a competitive advantage if the subproblem evaluations are not solved to global optimality, if evaluations are noisy and potentially inconsistent, or if they might change over time. This would be the case when objective evaluations are obtained by querying human decision-makers as opposed to optimization or simulation software. This would enable coordination between business units where some decision-makers still use expert-guided heuristics rather than numerical optimization.

AUTHOR CONTRIBUTIONS

Damien van de Berg: Formal analysis (lead); investigation (lead); methodology (equal); software (lead); visualization (lead); writing – original draft (supporting). **Panagiotis Petsagkourakis:** Methodology (equal); writing – review and editing (equal). **Nilay Shah:** Funding acquisition (equal); supervision (equal); writing – review and editing (equal). **Ehecatl Antonio del Rio-Chanona:** Conceptualization (lead); funding acquisition (equal); methodology (equal); supervision (equal); writing – review and editing (equal).

ACKNOWLEDGMENTS

D.v.d.B. gratefully acknowledges financial support from the EPSRC DTA studentship award and the Bansal bursary. D.v.d.B. would also

like to thank all BASF S.E. collaborators, especially his co-supervisors Dr. Debora Morgenstern, Dr. Daniel Engel, and Dr. Inga-Lena Darkow, scientific exchange coordinator Dr. Christian Holtze, and Dr. Sangbum Lee for their invaluable support in conceptualizing this work.

DATA AVAILABILITY STATEMENT

All data necessary for replication of results can be found with the code under https://github.com/OptiMaL-PSE-Lab/Data-driven-coordination.

ORCID

Damien van de Berg https://orcid.org/0000-0001-9167-5545
Panagiotis Petsagkourakis https://orcid.org/0000-0002-2024-3371
Ehecatl Antonio del Rio-Chanona https://orcid.org/0000-0003-0274-2852

REFERENCES

1. Gounaris CE, Grossmann IE. A preface to the special issue on enterprise-wide optimization. *Optim Eng*. 2019;20:965-968. doi:10.1007/s11081-019-09468-9
2. Grossmann I. Enterprise-wide optimization: a new frontier in process systems engineering. *AIChE J*. 2005;51(7):1846-1857. doi:10.1002/aic.10617
3. Grossmann IE. Advances in mathematical programming models for enterprise-wide optimization. *Comput Chem Eng*. 2012;47:2-18. doi:10.1016/J.COMPCHEMENG.2012.06.038
4. Chu Y, You F. Model-based integration of control and operations: overview, challenges, advances, and opportunities. *Comput Chem Eng*. 2015;83:2-20. doi:10.1016/j.compchemeng.2015.04.011
5. Bhosekar A, Ierapetritou M. Advances in surrogate based modeling, feasibility analysis, and optimization: a review. *Comput Chem Eng*. 2018;108:250-267. doi:10.1016/j.compchemeng.2017.09.017
6. Kim SH, Boukouvala F. Machine learning-based surrogate modeling for data-driven optimization: a comparison of subset selection for regression techniques. *Optim Lett*. 2020;14:989-1010. doi:10.1007/s11590-019-01428-7
7. Biegler LT, Lang YD, Lin W. Multi-scale optimization for process systems engineering. *Comput Chem Eng*. 2014;60:17-30. doi:10.1016/j.compchemeng.2013.07.009
8. Palomar DP, Chiang M. A tutorial on decomposition methods for network utility maximization. *IEEE J Sel Areas Commun*. 2006;24(8):1439-1451. doi:10.1109/JSAC.2006.879350
9. Iyer RR, Grossmann IE. A bilevel decomposition algorithm for long-range planning of process networks. *Ind Eng Chem Res*. 1998;37:474-481.
10. Dogan ME, Grossmann IE. A decomposition method for the simultaneous planning and scheduling of single-stage continuous multiproduct plants. *Ind Eng Chem Res*. 2006;45:299-315.
11. Pishvaee M, Razmi J, Torabi S. An accelerated benders decomposition algorithm for sustainable supply chain network design under uncertainty: a case study of medical needle and syringe supply chain. *Transp Res E: Logist Transp Rev*. 2014;67:14-38. doi:10.1016/j.tre.2014.04.001
12. Uster H, Easwaran G, Akcali E, Çetinkaya S. Benders decomposition with alternative multiple cuts for a multi-product closed-loop supply chain network design model. *Naval Res Logist (NRL)*. 2007;54:890-907. doi:10.1002/nav.20262
13. Santoso T, Ahmed S, Goetschalckx M, Shapiro A. A stochastic programming approach for supply chain network design under uncertainty. *Eur J Oper Res*. 2003;167:96-115. doi:10.18452/8297
14. Jackson JR, Grossmann IE. Temporal decomposition scheme for nonlinear multisite production planning and distribution models. *Ind Eng Chem Res*. 2003;42(13):3045-3055. doi:10.1021/ie030070p
15. Oliveira F, Gupta V, Hamacher S, Grossmann IE. A Lagrangean decomposition approach for oil supply chain investment planning under uncertainty with risk considerations. *Comput Chem Eng*. 2013;50:184-195.
16. Terrazas-Moreno S, Grossmann I. A multiscale decomposition method for the optimal planning and scheduling of multi-site continuous multiproduct plants. *Chem Eng Sci*. 2011;66:4307-4318. doi:10.1016/j.ces.2011.03.017
17. Sousa R, Liu S, Papageorgiou L, Shah N. Global supply chain planning for pharmaceuticals. *Chem Eng Res Des*. 2011;89:2396-2409. doi:10.1016/j.cherd.2011.04.005
18. Shin S, Hart P, Jahns T, Zavala VM. A hierarchical optimization architecture for large-scale power networks. *IEEE Trans Control Netw Syst*. 2019;6(3):1004-1014. doi:10.1109/TCNS.2019.2906917
19. Boyd S, Parikh N, Chu E, Peleato B, Eckstein J. *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers*. Now Foundations and Trends; 2010. http://www.nowpublishers.com/article/Details/MAL-016. doi:10.1561/2200000016
20. Rodriguez J, Nicholson B, Laird C, Zavala V. Benchmarking ADMM in nonconvex NLPS. *Comput Chem Eng*. 2018;119:315-325. doi:10.1016/j.compchemeng.2018.08.036
21. Buccini A, Dell'Acqua P, Donatelli M. A general framework for ADMM acceleration. *Numer Algorith*. 2020;85:829-848. doi:10.1007/s11075-019-00839-y
22. Houska B, Frasch J, Diehl M. An augmented Lagrangian based algorithm for distributed nonconvex optimization. *SIAM J Optim*. 2016;26(2):1101-1127. doi:10.1137/140975991
23. Fourer R, Ma J, Martin K. Optimization services: a framework for distributed optimization. *Oper Res*. 2010;58(6):1624-1636. http://www.jstor.org/stable/40984032
24. Tsianos KI, Lawlor S, Rabbat MG. Consensus-based distributed optimization: practical issues and applications in large-scale machine learning. *2012 50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. Allerton Park & Retreat Center; 2012:1543-1550. doi:10.1109/Allerton.2012.6483403
25. Yang T, Yi X, Wu J, et al. A survey of distributed optimization. *Annu Rev Control*. 2019;47:278-305. doi:10.1016/j.arcontrol.2019.05.006
26. Berahas AS, Cao L, Choromanski K, Scheinberg K. A theoretical and empirical comparison of gradient approximations in derivative-free optimization. *Found Comput Math*. 2019;22:507-560. https://doi.org/10.1007/s10208-021-09513-z
27. Herrera M, Pérez-Hernández M, Kumar Parlikad A, Izquierdo J. Multi-agent systems and complex networks: review and applications in systems engineering. *Processes*. 2020;8(3):312. doi:10.3390/pr8030312
28. Dominguez R, Cannella S. Insights on multi-agent systems applications for supply chain management. *Sustainability (Switzerland)*. 2020;12(5):1-13. www.mdpi.com/journal/sustainability. doi:10.3390/su12051935
29. Engelmann A, Jiang Y, Houska B, Faulwasser T. Decomposition of nonconvex optimization via bi-level distributed ALADIN. *IEEE Trans Control Netw Syst*. 2020;7(4):1848-1858. doi:10.1109/TCNS.2020.3005079
30. Larson J, Menickelly M, Wild SM. Derivative-free optimization methods. *Acta Numer*. 2019;28:287-404. doi:10.1017/S0962492919000060
31. Amaran S, Sahinidis NV, Sharda B, Bury SJ. Simulation optimization: a review of algorithms and applications. *Ann Oper Res*. 2016;240(1):351-380. doi:10.1007/s10479-015-2019-x
32. van de Berg D, Savage T, Petsagkourakis P, Zhang D, Shah N, del Rio-Chanona EA. Data-driven optimization for process systems engineering applications. *Chem Eng Sci*. 2022;248:117135. doi:10.1016/J.CES.2021.117135

33. Zhao F, Grossmann IE, García-Muñoz S, Stamatis SD. Flexibility index of black-box models with parameter uncertainty through derivative-free optimization. *AIChE J*. 2021;67:e17189. doi:10.1002/aic.17189
34. Beykal B, Avraamidou S, Pistikopoulos E. Bi-level mixed-integer data-driven optimization of integrated planning and scheduling problems. *Comput Chem Eng*. 2021;50. ISBN 9780323885065:1707-1713. doi:10.1016/B978-0-323-88506-5.50265-5
35. Beykal B, Avraamidou S, Pistikopoulos IP, Onel M, Pistikopoulos EN. DOMINO: data-driven optimization of bi-level mixed-integer nonlinear problems. *J Glob Optim*. 2020;78(1):1-36. doi:10.1007/s10898-020-00890-3
36. van de Berg D, Petsagkourakis P, Shah N, del Rio-Chanona EA. Data-driven distributed optimization for systems consisting of expensive black-box subproblems. *14th International Symposium on Process Systems Engineering* (accepted). 2022.
37. Tang W, Daoutidis P. Distributed control and optimization of process system networks: a review and perspective. *Chin J Chem Eng*. 2019;27(7):1461-1473. doi:10.1016/J.CJCHE.2018.08.027
38. Ghadimi E, Teixeira A, Shames I, Johansson M. Optimal parameter selection for the alternating direction method of multipliers (ADMM): quadratic problems. *IEEE Trans Autom Control*. 2014;60:644-658. doi:10.1109/TAC.2014.2354892
39. Xu Y, Liu M, Lin Q, Yang T. ADMM without a fixed penalty parameter: faster convergence with new adaptive penalization. *Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17*. Curran Associates Inc. ISBN 978-1-5108-6096-4; 2017:1267-1277.
40. Nishihara R, Lessard L, Recht B, Packard A, Jordan MI. A general analysis of the convergence of ADMM. *Int Conf Mach Learn*. 2015;32:343-352. abs/1502.02009.
41. Li Z, Dong Z, Liang Z, Ding Z. Surrogate-based distributed optimisation for expensive black-box functions. *Automatica*. 2021;125:109407. doi:10.1016/j.automatica.2020.109407
42. Boyd S, Vandenberghe L. *Convex Optimization*. Cambridge University Press; 2004. https://www.cambridge.org/core/books/convex-optimization/17D2FAA54F641A2F62C7CCD01DFA97C4. doi:10.1017/cbo9780511804441
43. Cartis C, Roberts L, Sheridan-Methven O. Escaping local minima with derivative-free methods: a numerical investigation. *Optimization*. 2019;71(8);2343-2373. doi:10.1080/02331934.2021.1883015
44. Cartis C, Roberts L, Sheridan-Methven O. Escaping local minima with local derivative-free methods: a numerical investigation. *Optimization*. 2021;71:2343-2373. doi:10.1080/02331934.2021.1883015
45. Shields BJ, Stevens J, Li J, et al. Bayesian reaction optimization as a tool for chemical synthesis. *Nature*. 2021;590:89-96. doi:10.1038/s41586-021-03213-y
46. Felton KC, Rittig JG, Lapkin AA. Summit: benchmarking machine learning methods for reaction optimisation. *Chemistry–Methods*. 2021;1(2):116-122. doi:10.1002/cmtd.202000051
47. Olofsson S, Schultz ES, Mhamdi A, Mitsos A, Deisenroth MP, Misener R. Design of Dynamic Experiments for Black-Box Model Discrimination. ArXiv. 2021. abs/2102.03782.
48. Chanona EAR, Petsagkourakis P, Bradford E, Graciano JE, Chachuat B. Real-time optimization meets Bayesian optimization and derivative-free optimization: a tale of modifier adaptation. *Comput Chem Eng*. 2021;147:107249. doi:10.1016/j.compchemeng.2021.107249
49. Paulson J, Makrigiorgos G, Mesbah A. Adversarially robust bayesian optimization for efficient auto-tuning of generic control structures under uncertainty. *AIChE J*. 2022;68:e17591. doi:10.1002/aic.17591
50. Makrigiorgos G, Bonzanini AD, Miller V, Mesbah A. Performance-oriented model learning for control via multi-objective Bayesian optimization. *Comput Chem Eng*. 2022;162:107770. doi:10.1016/j.compchemeng.2022.107770
51. The GPyOpt authors. *Gpyopt: A Bayesian Optimization Framework in Python*; 2016. http://github.com/SheffieldML/GPyOpt.
52. Garnett R. *Bayesian Optimization*. Cambridge University Press; 2022 In preparation.
53. Johnson SG. *The NLopt nonlinear-optimization package*; 2020. http://github.com/stevengj/nlopt.
54. Jones DR, Perttunen CD, Stuckman BE. Lipschitzian optimization without the Lipschitz constant. *J Optim Theory Appl*. 1993;79(1):157-181. doi:10.1007/BF00941892
55. Neumaier, A. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, 2004;13:271-369. doi:10.1017/S0962492904000194
56. Gablonsky JM, Kelley CT. A locally-biased form of the DIRECT algorithm. *J Glob Optim*. 2001;21(1):27-37. doi:10.1023/A:1017930332101
57. Hart WE, Watson JP, Woodruff DL. Pyomo: modeling and solving mathematical programs in python. *Math Program Comput*. 2011;3(3):219-260.
58. Bynum ML, Hackebeil GA, Hart WE, et al. *Pyomo–Optimization Modeling in Python*. Vol 67. 3rd ed. Springer Science & Business Media; 2021.
59. Wächter A, Biegler LT. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math Program*. 2006;106(1):25-57. doi:10.1007/s10107-004-0559-y
60. Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*; 2022. https://www.gurobi.com.
61. Fudenberg D, Tirole J. *Game Theory*. MIT Press; 1991 Translated into Chinesse by Renin University Press, Bejing: China.
62. Maestre J, Pena D, Camacho E. Distributed model predictive control based on a cooperative game. *Optim Control Appl Methods*. 2011;32:153-176. doi:10.1002/oca.940
63. Arauz Pison T, Chanfreut P, Maestre J. Cyber-security in networked and distributed model predictive control. *Annu Rev Control*. 2021;53:338-355. doi:10.1016/j.arcontrol.2021.10.005
64. Kairouz P, McMahan HB, Avent B, et al. Advances and open problems in federated learning. *Found Trends Mach Learn*. 2021;14(1–2):1-210. doi:10.1561/2200000083
65. Rodríguez-Barroso N, Stipcich G, Jiménez-López D, et al. Federated learning and differential privacy: software tools analysis, the sherpa.ai fl framework and methodological guidelines for preserving data privacy. *Inf Fusion*. 2020;64:270-292. doi:10.1016/j.inffus.2020.07.009
66. Yao AC. Protocols for secure computations. *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, Chicago, IL; 1982:160–164. 10.1109/SFCS.1982.38
67. Yang Q, Liu Y, Cheng Y, Kang Y, Chen T, Yu H. Federated Learning. Springer Nature. 2019. https://doi.org/10.1007/978-3-031-01585-4
68. McMahan HB, Moore E, Ramage D, Hampson S, Arcas BA. *Communication-Efficient Learning of Deep Networks from Decentralized Data*. AISTATS; 2017.
69. Wang Z, Zhang J, Chang TH, Li J, Luo ZQ. Distributed stochastic consensus optimization with momentum for nonconvex nonsmooth problems. *IEEE Trans Signal Process*. 2021;69:4486-4501. doi:10.1109/tsp.2021.3097211
70. Xu Y, Zhu S, Yang S, Zhang C, Jin R, Yang T Learning with non-convex truncated losses by SGD. *35th Conference on Uncertainty in Artificial Intelligence, UAI 2019*; 2018. https://arxiv.org/abs/1805.07880v1.
71. Shah N. Process industry supply chains: advances and challenges. *Comput Chem Eng*. 2005;29(6):1225-1235. doi:10.1016/j.compchemeng.2005.02.023 selected Papers Presented at the 14th European Symposium on Computer Aided Process Engineering.
72. Yue D, You F. Stackelberg-game-based modeling and optimization for supply chain design and operations: a mixed integer bilevel

programming framework. *Comput Chem Eng*. 2017;102:81-95. doi:10.1016/J.COMPCHEMENG.2016.07.026
73. Chen CL, Lee WC. Multi-objective optimization of multi-echelon supply chain networks with uncertain product demands and prices. *Comput Chem Eng*. 2004;28:1131-1144. doi:10.1016/j.compchemeng.2003.09.014
74. Zavala V. Chapter seven – Managing conflicts among decision-makers in multiobjective design and operations. In: Ruiz-Mercado G, Cabezas H, eds. *Sustainability in the Design, Synthesis and Analysis of Chemical Engineering Processes*. Butterworth-Heinemann. ISBN 978-0-12-802032-6; 2016:169-180. doi:10.1016/B978-0-12-802032-6.00007-4
75. Allman A, Zhang Q. Distributed fairness-guided optimization for coordinated demand response in multi-stakeholder process networks. *Comput Chem Eng*. 2022;161:107777. doi:10.1016/j.compchemeng.2022.107777
76. Krishnamoorthy D. A distributed feedback-based online process optimization framework for optimal resource sharing. *J Process Control*. 2021;97:72-83.
77. Sampat AM, Martin E, Martin M, Zavala VM. Optimization formulations for multi-product supply chain networks. *Comput Chem Eng*. 2017;104:296-310. doi:10.1016/j.compchemeng.2017.04.021
78. Garcia DJ, You F. The water-energy-food nexus and process systems engineering: a new focus. *Comput Chem Eng*. 2016;91:49-67. doi:10.1016/j.compchemeng.2016.03.003 12th International Symposium on Process Systems Engineering & 25th European Symposium of Computer Aided Process Engineering (PSE-2015/ESCAPE-25), 31 May–4 June 2015, Copenhagen, Denmark.
79. Shah N. Pharmaceutical supply chains: key issues and strategies for optimisation. *Comput Chem Eng*. 2004;28(6):929-941. doi:10.1016/j.compchemeng.2003.09.022 fOCAPO 2003 Special issue.
80. Sarkis M, Bernardi A, Shah N, Papathanasiou MM. Emerging challenges and opportunities in pharmaceutical manufacturing and distribution. *Processes*. 2021;9(3). https://www.mdpi.com/2227-9717/9/3/457:457. doi:10.3390/pr9030457
81. Lara CL, Trespalacios F, Grossmann IE. Global optimization algorithm for capacitated multi-facility continuous location-allocation problems. *J Glob Optim*. 2018;71(4):871-889. doi:10.1007/S10898-018-0621-6
82. Tables of composition and nutritional values of feed materials INRA CIRAD AFZ.
83. Boyd S, Xiao L, Mutapcic A. Notes on Decomposition Methods. Tech. Rep.; 2003.
84. Kouzoupis D, Quirynen R, Houska B, Diehl M. A block based ALADIN scheme for highly parallelizable direct optimal control. *Proceedings of the American Control Conference*. ISBN 9781467386821; 2016. 10.1109/ACC.2016.7525066

SUPPORTING INFORMATION

Additional supporting information can be found online in the Supporting Information section at the end of this article.

How to cite this article: van de Berg D, Petsagkourakis P, Shah N, del Rio-Chanona EA. Data-driven coordination of subproblems in enterprise-wide optimization under organizational considerations. *AIChE J*. 2023;69(4):e17977. doi:10.1002/aic.17977