Operationalizing Experimental Design: Data Collection for Remote Ocean Monitoring

Jean Pauphilet
Management Science and Operations, London Business School, London, United Kingdom
jpauphilet@london.edu

Baizhi Song
Management Science and Operations, London Business School, London, United Kingdom
bsong@london.edu

Problem definition: To collect data on ocean plastic pollution and build more accurate predictive models, we need to manually take high-resolution pictures of the sea surface via floating or flying drones. Operating these vehicles, like many data collection problems in agriculture or environmental science, challenges the traditional optimal experimental design (OED) formulation from statistics by its scale as well as the presence of complex routing constraints. *Methodology/results:* We develop a discrete optimization algorithm to solve large-scale instances of OED as well as account for routing constraints. On synthetic and real-world data, our algorithm outperforms existing solutions relying on commercial branch-and-bound solvers. For our problem of ocean plastic density prediction, for example, it finds solutions with up to 10% higher objective value and solves twice as many instances to optimality. In other words, it can build ensemble models with the same accuracy as a random data collection strategy, yet with a 30–50% lower cost. *Managerial implications:* Our study provides an efficient algorithm for solving large-scale instances of OED problems with routing constraints. It highlights the benefit of integrating operational constraints like routing into the design of data collection strategies to reduce data collection cost.

Key words: Optimal experimental design; Model ensemble; Binary optimization; Orienteering problem; Environment

1. Introduction

Despite the recent explosion in data availability, data collection in certain fields remains challenging and expensive. Applications of analytics to agriculture and environmental sciences, for instance, require the deployment of large networks of sensors, potentially in remote areas and under difficult operating conditions. These settings provide new large-scale examples of optimal experimental design (OED), a cornerstone problem in statistics (see, e.g., Kiefer 1959, Atkinson and Fedorov 1975, Fedorov 2010). OED refers to the problem selecting from a large population a small subset of individuals (or samples) that is as representative and informative of the entire population as possible—an intuition we formalize in Section 1.2. In particular, in this paper, we are interested in collecting high-resolution images of the ocean surface in order to build ensemble models for ocean

plastic density prediction (see Section 1.1). OED has numerous other applications, ranging from measurement-constrained regression (Wang et al. 2017) to sensor location (Joshi and Boyd 2008).

Because OED is an NP-hard combinatorial optimization problem (Welch 1982a), most of the algorithms that have been proposed for solving it rely on convex relaxations followed by local search techniques (e.g., Madan et al. 2019) or tailored sampling schemes (e.g., Singh and Xie 2020). Despite strong theoretical guarantees and empirical performance, these approaches are not able to handle complex operational requirements that might be relevant in practice. For example, for geospatial data, one might prefer to select samples that are physically close or along a given trajectory, e.g., using drones and other unmanned vehicles. Such routing constraints limit how far apart the sampled locations can be from each other, thus impeding the OED objective, which tends to favor sampling from diverse locations. In this context, formal discrete optimization methods provide the modeling flexibility needed to incorporate complex operational constraints.

Our objective in this paper is to develop an optimization-based strategy to efficiently collect data about ocean plastic pollution (see Section 1.1 for a detailed description of our target use case). Specifically, we develop and validate a tailored branch-and-bound algorithm for solving OED with routing constraints. To do so, we first consider the classical D-optimal experimental design problem from the statistics literature (without routing) and design a comprehensive discrete optimization strategy, which largely outperforms commercial solvers. We then extend each ingredient of this strategy to incorporate routing constraints. Our work explores the exciting intersection of data mining and transportation problems, and illustrates the power of discrete optimization to bring statistical learning objectives and operational requirements together.

1.1. Use case: Data collection in remote maritime regions

We are collaborating with a non-governmental organization (NGO) that aims to reduce floating plastic pollution in the oceans. They have developed a plastic collection system, consisting of two boats and a large interception barrier, that collects floating plastic and have been operating in the Great Pacific Garbage Patch (GPGP; an area twice the size of Texas and located halfway between Hawaii and California) since 2021. In order to inform their operations, they have developed several models to predict future locations and movements of plastic pieces.

Unfortunately, pieces of plastic that float in the GPGP cannot be observed from satellite images. So, the only solution to gather data on this issue is to physically go there and take samples or pictures. For this reason, to this day, there is no reliable data providing systematic measurements of plastic concentrations with a broad spatial and temporal coverage. Given the scarcity of data, scientists

have been primarily relying on physics-based modeling to describe the dispersion dynamics of free-floating plastic and estimate plastic concentrations (see, e.g., Kaandorp et al. 2023). These models take as primary inputs sea currents and wind information (direction and speed).

However, even for the same input (e.g., sea current), there are discrepancies between data coming from different providers. Moreover, some parameter values (e.g., the contribution of wind speed to plastic speed) are not precisely known. Hence, our partner NGO has a collection of plastic dispersion models available, corresponding to different data sources and different model parameters. As a result, for each location in the GPGP (and time point), they have not one but several plastic density estimates. In this context, they are interested in collecting ground truth data (e.g., by sending flying or floating drones equipped with high-resolution cameras) to improve their prediction accuracy.

In the short/medium term, it is unrealistic to assume that they could collect enough data to stop using physics-based models and rely on data-driven models only. However, there is an opportunity to improve predictive accuracy by ensembling the existing models together. Formally, each location is currently associated with a vector $\boldsymbol{x} \in \mathbb{R}^p$ corresponding to the plastic density estimates generated using p different physics-based models. Given observations about the actual plastic density y, constructing an ensemble model corresponds to constructing a linear combination of the observable inputs, $\boldsymbol{\beta}^\top \boldsymbol{x}$, that best predicts y. While most of the literature on model ensembling considers the problem of learning the weights $\boldsymbol{\beta}$ (also know as super ensemble in the weather forecasting literature, Krishnamurti et al. 1999, Kharin and Zwiers 2002, Krishnamurti et al. 2016), we are interested in the upstream problem of collecting ground truth labels to enable such ensembling rules to be learned.

Remark 1. Physics-based models are usually good at predicting short-term transitions. However, they often rely on long-range simulations, where error can accumulate and decalibrate the models. In practice, we envision using ensembling to obtain a better estimate of plastic density in the GPGP at a given point in time, and generate estimates on future plastic density by applying the physics-based models with this new initialization (see also the related literature on data assimilation, e.g., Bach and Ghil 2023, and references therein).

1.2. Optimal Experimental Design (OED): Motivation and formulation

The problem of collecting data for learning how to ensemble models falls within the scope of OED for linear regression: Assuming data is generated according to an underlying linear model of the form $y = \boldsymbol{x}^\top \boldsymbol{\beta}^\star + \epsilon$, where $\boldsymbol{x} \in \mathbb{R}^p$ is a vector of observed covariates and ϵ a white noise. We can estimate $\boldsymbol{\beta}^\star$ from n independent observations $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, n$, via, among others, the

ordinary least square (OLS) estimator $\hat{\boldsymbol{\beta}}_{OLS} := (\boldsymbol{X}^\top \boldsymbol{X})^{-1}\boldsymbol{X}^\top \boldsymbol{y}$ where $\boldsymbol{X} \in \mathbb{R}^{n\times p}$ is a matrix whose rows are the individual observations $\boldsymbol{x}_i$'s. If the errors are independent and normally distributed, $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, then $\hat{\boldsymbol{\beta}}_{OLS} \sim \mathcal{N}\left(\boldsymbol{\beta}^\star, \sigma^2(\boldsymbol{X}^\top \boldsymbol{X})^{-1}\right)$. Hence, the matrix $(\boldsymbol{X}^\top \boldsymbol{X})^{-1}$ directly controls the estimation error $\|\hat{\boldsymbol{\beta}}_{OLS} - \boldsymbol{\beta}^\star\|^2$.

In OED, the objective is to select some observations $\boldsymbol{x}_i$ *without knowledge of the signals* y_i (observation of y_i will occur after selection) in order to allow for the 'best' downstream estimation. Denoting $z_i = 1$ if we decide to observe sample i, and $z_i = 0$ otherwise, a good criterion is to minimize the variance of the resulting OLS estimator, in other words, to 'minimize' the matrix $(\boldsymbol{X}^\top \operatorname{diag}(\boldsymbol{z})\boldsymbol{X})^{-1}$. Different metrics have been proposed to quantify the 'size' of $(\boldsymbol{X}^\top \operatorname{diag}(\boldsymbol{z})\boldsymbol{X})^{-1}$. Although most criteria involve the information matrix, $\boldsymbol{X}^\top \operatorname{diag}(\boldsymbol{z})\boldsymbol{X}$, or its inverse, the relevance of a specific criterion usually depends on the downstream statistical task to be performed. For example, the T-optimal design problem focuses on maximizing the ability to discriminate between two models (Atkinson and Fedorov 1975). We refer to Pukelsheim (2006), Fedorov (2010) for reviews and to Wong (1994) for numerical comparison of the solutions selected by different criteria.

In this paper, we focus our attention to the solution of the canonical binary D-optimal experimental design (OED) problem

$$\max_{\boldsymbol{z}\in\{0,1\}^n} \quad \operatorname{logdet}(\boldsymbol{X}^\top \operatorname{diag}(\boldsymbol{z})\boldsymbol{X}) \quad \text{s.t.} \quad \sum_{i=1}^{n} z_i \le d, \tag{1}$$

where $p \le d \le n$, although our approach could be generalized to a broader set of optimality criteria (see Remark EC.1). The binary decision space $\boldsymbol{z} \in \{0,1\}^n$ in (1) implies that each observation can only be selected once. The OED problem with repetitions relaxes this assumption by allowing $\boldsymbol{z}$ to be any non-negative integer (up to a limit k), which can be reduced to a problem of the form (1) with kn observations by duplicating the $\boldsymbol{x}_i$'s. In some works, design optimization is understood as optimizing for $\boldsymbol{x}_i$ directly (within a continuous design space) instead of picking from a discrete set of values (Hoel 1958).

Formulation (1) is derived after assuming a linear generating model, $\boldsymbol{x}^\top \boldsymbol{\beta}^\star$, with independent identically distributed (i.i.d.) noise terms ϵ_i. Alternative formulations have been proposed to relax both assumptions. For example, Fedorov and Leonov (chapters 5-6, 2013) present extension of the OED problem to non-linear response models, which usually invoke an estimate of $\beta^\star$ to locally approximate the variance matrix (see, e.g., Chernoff 1953, Yang et al. 2013). When noise terms cannot be assumed i.i.d., like in spatio-temporal models, approaches include redefining the

variance matrix to depend on the matrix of error correlations (Bischoff 1992, Dette et al. 2015) or including additional independent variables such as spatial and temporal features (thus increasing the dimension p) to ensure that the residual error terms satisfy the i.i.d. assumption. Alternatively, robust experimental design seeks to find designs that perform well under model misspecification (Box and Draper 1959, DuMouchel and Jones 1994, Goos et al. 2005). In case of extreme misspecification actually, randomized designs are considered a sensible strategy (Wu 1981, Waite and Woods 2022). In this paper, we are primarily interested in the intersection of OED with routing constraints so we focus our analysis on the canonical formulation (1), though our algorithm can be extended to more sophisticated settings in future research. We will also numerically evaluate the robustness of our solution when the linearity and independent noise assumptions are violated.

In practice, the fact that the data is collected via vehicles, however, imposes new and conflicting constraints for the standard OED problem. Given the spatial correlations in plastic density, Formulation (1) would recommend collecting data from locations that are far from one another. On the other hand, the requirement that these locations are part of the same trajectory favors data collection from neighboring locations. In this work, we resolve this trade-off by using formal optimization methods to explicitly account for routing constraints.

1.3. Contributions and structure

In this paper, we consider a D-optimal experimental design problem with routing constraints

$$\max_{z \in \mathcal{Z}} \quad \operatorname{logdet}(\boldsymbol{X}^\top \operatorname{diag}(z) \boldsymbol{X}), \tag{2}$$

where the set $\mathcal{Z} \subseteq \{0,1\}^n$ enforces that the sampled observations $\{i : z_i = 1\}$ correspond to a trajectory starting and ending at a given depot. We motivated this problem in the context of collecting environmental data in remote marine environments in Section 1.1, but it has also been found relevant in the context of post-disaster assessment (Wang et al. 2025). The emergence of an information criterion like D-optimality into a classical vehicle routing setting, in two very different use cases, highlights the relevance of the frontier between machine learning and transportation science.

Our main objective is the comprehensive development of a branch-and-bound algorithm to solve this new class of problems at scale, and its validation on our motivating case study: building accurate ensemble models for ocean plastic prediction. The rest of the paper is organized as follows.

After reviewing the relevant literature in Section 2, we start by presenting our branch-and-bound algorithm in the case of the classical OED problem (1), without routing constraints, in Section 3. This

simpler setting allows us to introduce the generic blueprint for our branch-and-bound algorithm, which we later generalize to account for routing constraints. A core idea in our algorithm is to use an efficient first-order algorithm (here, Frank-Wolfe) to solve each node relaxation, without the need to make any approximation of the log-determinant objective (unlike alternative outer-approximation strategies). Other important components are screening rules, to fix some coordinates to 0 or 1 a priori and efficiently reduce the problem dimension.

This detour is not only pedagogic, but also practically meaningful. Indeed, as we demonstrate numerically in Section 4, our branch-and-bound algorithm significantly outperforms commercial solvers for solving Problem (1), which is noteworthy for such a longstanding problem. Our algorithm solves to optimality four times more instances (60% of the instances in our testbed vs. 14%) and reduces computational time by 1–2 orders of magnitude. For example, it solves instances with $n = 10,000$ under a minute when traditional approaches reach this time limit for $n = 200$ only.

We then generalize the algorithmic ingredients of our branch-and-bound algorithm to the case with orienteering constraints (Gunawan et al. 2016, Vansteenwegen and Gunawan 2019) in Section 5. In particular, we similarly rely on Frank-Wolfe for node relaxations and derive screening rules based on the tour length constraint. While previous approaches enforce the routing requirements either via an exponential number of constraints (i.e., subtour elimination constraints as in Dantzig et al. 1954) or using generic integer variables (as in Miller et al. 1960), our solution is notable in that it leverages both formulations at different steps in the algorithm.

We demonstrate the computational benefits of our algorithm and its impact on the problem of creating accurate ocean model ensembles in Section 6. Our algorithm solves more than 40% of the instances under a minute (vs. 20% for existing approaches) and achieves a 10% optimality gap or lower on all instances (vs. less than a third of the instances). Importantly, for the downstream prediction problem of our partner NGO, we show that collecting data according to the solution of (2) outperforms alternative strategies, such as random designs, even in highly misspecified settings. Quantitatively, our approach constructs accurate predictive models with 30–50% less samples than alternatives, hence demonstrating the benefit of optimization on data collection cost.

Notations: We use lower-case (x), lower-case bold ($\boldsymbol{x}$), and upper-case bold characters ($\boldsymbol{X}$) to denote scalars, vectors, and matrices respectively. For the matrix $\boldsymbol{X} \in \mathbb{R}^{n\times p}$, we denote $\boldsymbol{x}_i \in \mathbb{R}^p$ is the i^{th} row of $\boldsymbol{X}$, i.e., $\boldsymbol{X}^\top = [\boldsymbol{x}_1 \cdots \boldsymbol{x}_n]$. We let $\boldsymbol{e}$ (resp. $\boldsymbol{0}$) denote the vector of all ones (resp. zeros), with dimension implied by the context. For matrix $\boldsymbol{X}$ and $\boldsymbol{Y}$, we use the notation $\boldsymbol{X} \succ \boldsymbol{Y}$ to indicate $\boldsymbol{X} - \boldsymbol{Y}$ being positive definite. We denote S_+^n as the set of all positive semidefinite $n \times n$ matrices.

2. Literature Review

Our work sits at the intersection of the optimal experimental design and vehicle routing literature.

Optimal Experimental Design (OED). Given the nature of our contribution, we focus our review on the algorithmic aspects of the D-optimal design literature, and refer to Atkinson (1996), Pukelsheim (2006), López-Fidalgo (2023) for a review of alternative problem formulations. Motivated by worst-case complexity results (Welch 1982a), polynomial-time algorithms that solve OED problems with constant-factor guarantees have been an active stream of research. One approach consists of obtaining a fractional solution by solving a continuous relaxation and then applying rounding or sampling techniques to get a feasible binary solution (Bouhtou et al. 2010, Wang et al. 2017, Allen-Zhu et al. 2017, Singh and Xie 2020, Lau and Zhou 2020, Nikolov et al. 2022). Another family of methods start from a feasible solution and improves it via local search (e.g., swaps, see Nguyen and Miller 1992, Fedorov 2010, Madan et al. 2019, Lau and Zhou 2022). However, even slight changes in the constraint of (1) necessitate substantial changes in these local search procedures (see, e.g., Wang et al. 2024, for OED with a generic budget constraint).

Initial attempts at developing certifiably optimal methods (i.e., branch-and-bound) for OED focused on obtaining valid upper bounds on the objective value of (1) without solving a semidefinite optimization problem, by leveraging Hadamard's inequality (Welch 1982b) or bounds based on singular values (Ko et al. 1994). Based on these bounds, Ponte et al. (2023) implement a branch-and-bound algorithm for OED with repetitions (i.e., where z can be any integer, not necessarily binary). Combined with efficient implementation of local search and variable tightening techniques, they solve instances with n up to 80, $p = n/4$, and $k/p = 2$. Sagnol and Harman (2015) deviate from the log-determinant objective and consider second-order-cone-representable objectives, leading to OED problems solvable using mixed-integer second-order-cone commercial solvers. For a similar class of objective functions, Ahipasaoglu (2021) designs a branch-and-bound algorithm with a coordinate-ascent strategy for solving the relaxations, and solves problems with $n = 50$–100, $p = 10$ within seconds.

OED with a log-determinant objective is closely related to data fusion (Li et al. 2024) and matrix entropy sampling (Anstreicher 2020, Li and Xie 2024), which are equivalent. The D-optimal data fusion problem maximizes the log determinant of $\boldsymbol{C} + \sum_i z_i \boldsymbol{x}_i \boldsymbol{x}_i^\top$ where $\boldsymbol{C} \succ \boldsymbol{0}$ is a given Fisher information matrix. Li et al. (2024) extend the local search and sampling technique of Singh and Xie (2020) to obtain constant-factor approximation algorithms for data fusion problems, and design

an outer approximation procedure (leveraging concavity and submodularity of the objective) to solve them to optimality.

The branch-and-bound algorithm we develop in this paper relies on efficiently solving semidefinite relaxations of (1) via first-order methods (Frank-Wolfe algorithm to be precise). The benefits compared to the literature are twofold: First, we make no approximation of the objective function, leading to tighter bounds (unlike methods that rely on Hadamard's inequality, spectral bounds, or piece-wise linear outer approximations). Second, the first-order method can be efficiently warm-started from one subproblem to another. We also develop new screening rules that can fix many binary variables at the root node. Thanks to these algorithmic improvements, our algorithm outperforms generalized Benders' (or outer approximation) procedures implemented using commercial solvers, echoing recent achievements of tailored branch-and-bound methods for machine learning (e.g., Hazimeh et al. 2022).

Vehicle Routing. We are interested in the extension of OED where the collected samples belong to a route. The problem of visiting a subset of locations to maximize a given reward under a tour length constraint is referred as the selective Traveling Salesperson Problem (selective TSP; Laporte and Martello 1990, Gendreau et al. 1998) or Orienteering Problem (OP; see Gunawan et al. 2016, Vansteenwegen and Gunawan 2019, for reviews). While early work on OP focused on heuristics methods (see Keller 1989, for a summary), several exact methods have been considered, starting with the work of Laporte and Martello (1990). Among others, Leifer and Rosenwein (1994) tighten OP's linear relaxation with valid linear cuts, and Bauer et al. (2002) derive facet-defining inequalities for the problem. Gendreau et al. (1998) develop a branch-and-price method.

Compared with the classical OP literature, we incorporate an information criterion as a direct objective of the routing problem. In particular, this objective cannot be decomposed as a sum over the visited locations (or edges). To the best of our knowledge, the work by Wang et al. (2025) and ours are the only ones to consider the combination of an explicit data collection objective with OP constraints. In terms of algorithm, Wang et al. (2025) rely on a commercial branch-and-bound code, using column and row generation procedures to iteratively introduce constraints (stemming from the piece-wise linear approximation of the objective and the subtour elimination constraints in the OP formulation), similar to the outer approximation algorithm we use as a benchmark. They apply their method to instances with $n = 50$–200 and $p = 14$. Our algorithmic strategy, on the other hand, perfectly replicates the one we develop for (1): We do not rely on any approximation of the log-determinant objective and solve the semidefinite relaxations via first-order methods by

leveraging new compact formulations of the OP constraints. Combined with screening rules and 'rounding' heuristics, our algorithms solve twice as many instances as commercial solvers and scale to instances with n in the $2,000$s.

3. An Efficient Branch-and-Bound Algorithm for OED

In this section, we propose a tailored branch-and-bound algorithm to solve (1) to provable optimality, which would serve as the blueprint for the case with routing constraints (Problem 2) in the rest of the paper. We first describe the current state-of-practice, namely outer approximation, in Section 3.1. We then present our branch-and-bound scheme (Section 3.2), which combines (i) an efficient first-order algorithm for solving node relaxations, (ii) fast heuristics for generating good feasible solutions, and (iii) novel screening rule (formally derived in Section 3.3).

3.1. Benchmark: Outer approximation

Problem (1) is a mixed-integer semidefinite optimization problem. Unfortunately, the technology for such problem class is not as mature as that for mixed-integer linear optimization. Hence, to the best of our knowledge, the most common approach for solving (1) exactly is based on linear outer approximation (Duran and Grossmann 1986). We first reformulate Problem (1) in epigraph formulation:

$$\max_{z\in\{0,1\}^n \boldsymbol{e}^\top z\le d} \quad \eta \quad \text{s.t} \quad \eta \le f(z). \tag{3}$$

Observe that $f(z) = \log\det(\boldsymbol{X}^\top \operatorname{diag}(z)\boldsymbol{X})$ is a concave function (see Lemma EC.2). Therefore, we can solve Problem (3) via outer approximation, i.e., by replacing the mixed-integer convex constraint $\eta \le f(z)$ by a set of linear constraints of the form $\eta \le f(z^t) + \langle \nabla f(z^t), z - z^t\rangle$, and iteratively adding linearization points z^t to refine this approximation. This scheme, which is referred to as a generalized Benders' decomposition algorithm, can be implemented with a commercial solver via lazy constraints and has been effectively used to solve large-scale statistical learning problems under sparsity (Bertsimas et al. 2020b,a, 2022). Li and Xie (2024), Li et al. (2024), Wang et al. (2025) use a similar scheme in the context of maximum entropy sampling, D-optimal data fusion, and D-optimal design respectively.

The benefit of outer approximation is to decompose the two difficulties in (1), namely the discrete and the semidefinite nature of the problem. The master problem solves a mixed-integer linear optimization problem at each iteration, while the semidefinite nature of the objective is deferred to the separation (i.e., cut generation) phase. Actually, the semidefinite optimization problem involved in separation can be solved in closed form via a simple matrix inversion (see Lemma EC.1).

On the negative side, outer approximation can be inefficient because it operates with an approximation of the log-determinant. When solving the Boolean relaxation in a node of the branch-and-bound tree, for example, the upper bound obtained combines the error from relaxing the integrality constraints and that from maximizing an approximation of the objective function. In the rest of this section, we develop a tailored branch-and-bound algorithm that can handle the log-determinant objective by design and does not suffer from any approximation error. As we will demonstrate, this procedure provides significant speed-ups compared with outer approximation (Section 4) and we generalize it to handle routing constraints in Sections 5–6.

3.2. Algorithm: Tailored branch-and-bound

Instead of relying on outer approximation and commercial branch-and-bound code, we propose in this section to design a tailored branch-and-bound algorithm for solving (1) to provable optimality.

Given two sets $\mathcal{I}_0, \mathcal{I}_1 \subseteq \{1, \dots, n\}$, we define the sub-problem $\mathcal{P}(\mathcal{I}_0, \mathcal{I}_1)$ as

$$\mathcal{P}(\mathcal{I}_0, \mathcal{I}_1) : \quad \max_{z \in \{0,1\}^n : z^\top \boldsymbol{e} \leq d} \operatorname{logdet}(\boldsymbol{X}^\top \operatorname{diag}(z) \boldsymbol{X}) \text{ s.t. } z_i = 0 \text{ for } i \in \mathcal{I}_0,\ z_i = 1 \text{ for } i \in \mathcal{I}_1.$$

Our branch-and-bound algorithm (described in Algorithm EC.4.1) maintains a list of active nodes, where each `node` corresponds to a subproblem $\mathcal{P}(\mathcal{I}_0, \mathcal{I}_1)$ and is associated with an upper and lower bound, denoted `node.ub` and `node.lb` respectively.

For each subproblem, we obtain `node.ub` by by replacing the integrality constraints $z \in \{0,1\}^n$ by simple box constraints $z \in [0,1]^n$ and solving the resulting semidefinite optimization problem using Frank-Wolfe algorithm, as suggested in the theoretical literature (see, e.g., Zhao and Freund 2023) and confirmed on preliminary numerical experiments (see Section EC.2). For the lower bound `node.lb`, we obtain a feasible solution by greedily rounding the solution of the Boolean relaxation $\bar{z}$, i.e., rounding to 1 its d largest coordinates and 0 the other ones. This greedy procedure, while common in mixed-integer literature (e.g., in Bertsimas et al. 2021), is not as popular for OED as sampling schemes (Singh and Xie 2020) or local search (Madan et al. 2019). However, preliminary numerical experiments suggest it produces competitive solutions and is more scalable (see Section EC.3).

In addition, at the root node, (i) we improve the quality of the initial feasible solution by conducting a local search (see Madan et al. 2019); (ii) we propose a screening rule to fix some of the variables and reduce the problem dimension. We formally describe and prove the validity of the screening rule in the following section.

3.3. Screening rule

Modern examples of OED involve a total population size n in the order of 10^4 or more. Hence, efficient screening rules to fix some of the coordinates of $\boldsymbol{z} \in \{0,1\}^n$ to 0 or 1 without loss of optimality can help increase the size of instances we can solve. For any positive definite matrix $\boldsymbol{A} \succ 0$, we can approximate the log-determinant objective linearly as follows

$$f(\boldsymbol{z}) = \operatorname{logdet}\left(\boldsymbol{X}^\top \operatorname{diag}(\boldsymbol{z})\boldsymbol{X}\right) \leq \sum_{i=1}^{n} \left(\boldsymbol{x}_i^\top \boldsymbol{A}\boldsymbol{x}_i\right) z_i - \operatorname{logdet}(\boldsymbol{A}) - p;$$

see Lemma EC.1 for a proof. We leverage this linear upper bound in our screening rule.

Proposition 1. *Fix $\boldsymbol{A} \succ 0$ and consider the associated linear upper approximation of $f(\boldsymbol{z})$*

$$f(\boldsymbol{z}) \leq \sum_{i=1}^{n} \left(\boldsymbol{x}_i^\top \boldsymbol{A}\boldsymbol{x}_i\right) z_i - \operatorname{logdet}(\boldsymbol{A}) - p =: \bar{f} + \sum_{j=1}^{d} \delta_{[j]}(z_{[j]} - 1) + \sum_{j=d+1}^{n} \delta_{[j]} z_{[j]},$$

where we denote $\delta_i := \boldsymbol{x}_i^\top \boldsymbol{A}\boldsymbol{x}_i$, $[i]$ the index of the i-th largest coordinate of $\boldsymbol{\delta}$, and $\bar{f} := \sum_{j=1}^{d} \delta_{[j]} - \operatorname{logdet}(\boldsymbol{A}) - p$. Further assume that we have a valid lower bound, LB, on the objective value of (1).

Then,

(i) if $\delta_i \geq \delta_{[d]}$ and $\delta_{[d+1]} - \delta_i < LB - \bar{f}$, we must have $z_i = 1$ at any optimal solution;

(ii) if $\delta_i \leq \delta_{[d+1]}$ and $\delta_i - \delta_{[d]} < LB - \bar{f}$, we must have $z_i = 0$ at any optimal solution.

Proof of Proposition 1 Observe that $\boldsymbol{\delta} \geq 0$ and that, for any $\boldsymbol{z}$,

$$f(\boldsymbol{z}) \;\leq\; \bar{f} + \sum_{j=1}^{d} \delta_{[j]}(z_{[j]} - 1) + \sum_{j=d+1}^{n} \delta_{[j]} z_{[j]}.$$

(i) Consider an index i such that $\delta_i \geq \delta_{[d]}$, i.e., $i = [j_0]$ for some $j_0 \leq d$. Then,

$$\max_{\boldsymbol{z}: z_i = 0} f(\boldsymbol{z}) \leq \bar{f} + \max_{\boldsymbol{z}: \boldsymbol{e}^\top \boldsymbol{z} = d,\, z_i = 0} \left\{ \sum_{j=1}^{d} \delta_{[j]}(z_{[j]} - 1) + \sum_{j=d+1}^{n} \delta_{[j]} z_{[j]} \right\} = \bar{f} + \delta_{[d+1]} - \delta_i.$$

Consequently, if $\bar{f} + \delta_{[d+1]} - \delta_i < LB$, then $\{\boldsymbol{z} : \boldsymbol{e}^\top \boldsymbol{z} = d,\, z_i = 0\}$ cannot contain any solution that achieves a higher objective value than LB, hence we can set $z_i = 1$ without loss of optimality.

(ii) Consider an index i such that $\delta_i \leq \delta_{[d+1]}$, i.e., $i = [j_0]$ for some $j_0 \geq d+1$. Then,

$$\max_{\boldsymbol{z}: \boldsymbol{e}^\top \boldsymbol{z} = d,\, z_i = 1} f(\boldsymbol{z}) \leq \bar{f} + \max_{\boldsymbol{z}: \boldsymbol{e}^\top \boldsymbol{z} = d,\, z_i = 1} \left\{ \sum_{j=1}^{d} \delta_{[j]}(z_{[j]} - 1) + \sum_{j=d+1}^{n} \delta_{[j]} z_{[j]} \right\} = \bar{f} + \delta_i - \delta_{[d]}.$$

Consequently, if $\bar{f} + \delta_i - \delta_{[d]} < LB$, then $\{\boldsymbol{z} : \boldsymbol{e}^\top \boldsymbol{z} = d,\, z_i = 1\}$ cannot contain any solution that achieves a higher objective value than LB, hence we can set $z_i = 0$ without loss of optimality. □

In practice, we apply the screening rule after solving the relaxation. Given a primal solution of the Boolean relaxation $\tilde{\boldsymbol{z}} \in [0,1]^n$, we apply Proposition 1 with $\tilde{\boldsymbol{A}} = (\boldsymbol{X}^\top \operatorname{diag}(\tilde{\boldsymbol{z}})\boldsymbol{X})^{-1} \succ \boldsymbol{0}$.

4. Validation on Synthetic Optimal Experimental Design Instances

In this section, we evaluate the effectiveness and relevance of our approach to synthetically generated instances of the classical OED problem (1).

4.1. Experimental setting

We generate data using the methodology in Wang et al. (2017). Observations $\boldsymbol{x}_i$ ($i = 1, \ldots, n$) are sampled independently from a multivariate Gaussian distribution $\mathcal{N}_p(\mathbf{0}, \boldsymbol{\Sigma}_0)$, where $\boldsymbol{\Sigma}_0$ is a skewed covariance matrix. Specifically, we construct $\boldsymbol{\Sigma}_0$ as $\boldsymbol{\Sigma}_0 = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\top$ where $\boldsymbol{U} \in \mathbb{R}^{p\times p}$ is a random orthogonal matrix and $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_p)$ with $\lambda_i = i^{-2}$ as Wang et al. (2017).

We generate instances for different values of n, p, and d, with n ranging from 150 to 10,000, p ranging from 5 to 60, and d/p ranging from 1 to 10. Compared to the scales considered in the existing literature, we consider values of n up one order of magnitude larger (10,000 vs 1,000–3,000 in Allen-Zhu et al. 2017, Wang et al. 2017, Li et al. 2024). Experiments are implemented in Julia and run on an Intel Xeon E5-2690 v4 2.6GHz CPU core with 36GB of memory.

4.2. Screening rule

We analyze how the screening rule derived in Section 3.3 performs numerically.

We first consider OED instances with $n = 1,000$, $p \in \{5, 10, 15, 20, 25\}$, and $d = 4 \times p$. For each instance, we solve the Boolean relaxation using FW algorithm with different levels of precision (optimality gap set to 10^{-2}, 10^{-3}, 10^{-4} respectively). We then apply the screening rule described in Proposition 1 and count the fraction of coordinates of $\boldsymbol{z}$ that can be fixed (either to 0 or 1). For each value of p, we average the performance across 10 replications. Results are presented in Figure 1. First, we observe that solving the relaxation accurately significantly impacts the performance of the screening rule. For $p = 15$, we can fix nearly 59% of the coordinates when solving the relaxation to 10^{-4}-optimality compared to 31% with 10^{-2}-optimality. Second, we observe that the screening rule is extremely powerful for $p = 5$–15 but becomes less effective as p increases.

Regarding the first observation, the benefit of solving the relaxation to higher accuracy can be twofold: a tighter relaxation can provide a better linear upper approximation of the objective (i.e., the matrix $\boldsymbol{A}$ in Proposition 1) as well as help generate a higher quality solution (i.e., LB). We conduct additional experiments to disentangle the contribution of each mechanism in Section EC.4.2 and find that the benefit mostly comes from the improved upper bound, the feasible solution heuristic (greedy rounding followed by local search) being quite robust to inaccuracies in the relaxation.

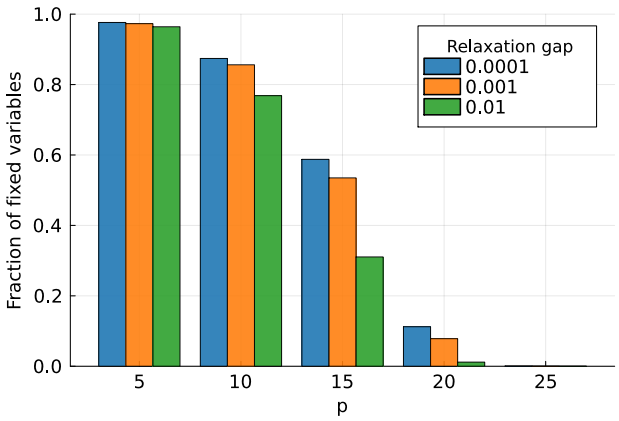


Figure 1 **Fraction of entries fixed (either to 0 or 1) by the screening rule, over 10 OED instances with $n = 1,000$, $d = 4 \times p$ and increasing p.**

4.3. Exact methods for achieving provable optimality

We evaluate the scalability of our tailored branch-and-bound as n, p, and d/p vary. To do so, we generate OED instances with $n = 1000$, $p = 20$, and $d/p = 4$, and, from this parameter combination, we vary each dimension separately (n, p, or p/d), the other two kept constant. Precisely, we generate various OED instances with $n \in \{100, 200, 500, 1000, 2000, 5000, 10000\}$, $p = 20$, and $d/p = 4$; $n = 1000$, $p \in \{5, 10, 15, 20, 40, 50, 60\}$, and $d/p = 4$; and $n = 1000$, $p = 20$, and $d/p \in \{1, 2, 3, 4, 5, 7, 10\}$. We generate 10 random instances for each parameter combination. We use a 60-second time limit and 0.1% optimality gap.

As presented in Section 3.1, we compare our branch-and-bound code to an outer-approximation scheme implemented with lazy callbacks and the MIO solver CPLEX 22.1. We accelerate the convergence of outer approximation by providing (i) the linear cut generated at the relaxed solution (warmstarting the upper bound) and (ii) the solution obtained by greedily rounding the solution of the relaxation followed by local search (warmstarting the lower bound). Indeed, in separate experiments (see Section EC.4.1), we find that this variant was the most effective at solving (1), and that CPLEX was marginally faster than Gurobi as a commercial solver.

Figure 2 represents the cumulative distribution of computational time (left panel) and optimality gap at termination (right panel) for the outer-approximation scheme and our branch-and-bound algorithm (Algorithm EC.4.1). Overall, we observe that outer approximation exhibits a sort of all-or-nothing behavior: It solves 10% of the instances in less than a second and 14% under a minute. On the remaining 86%, however, it fails to terminate within the time limit (60 seconds). Our tailored branch-and-bound, on the other hand, solves around 60% of the instances. Furthermore, when

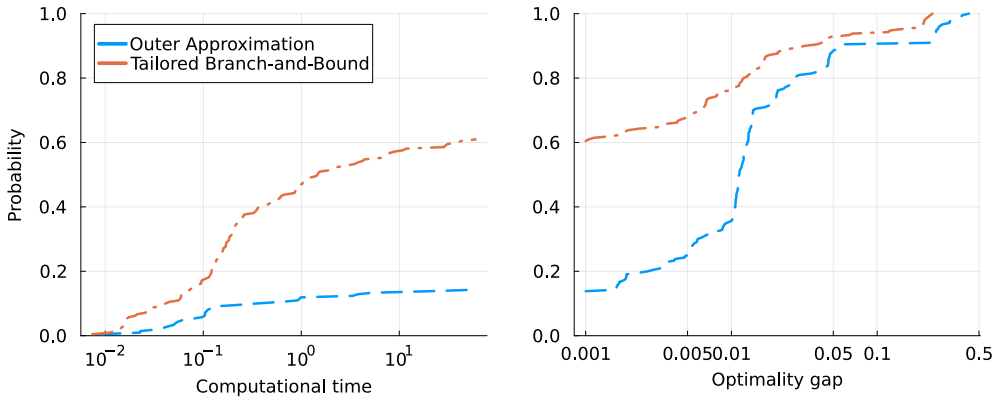


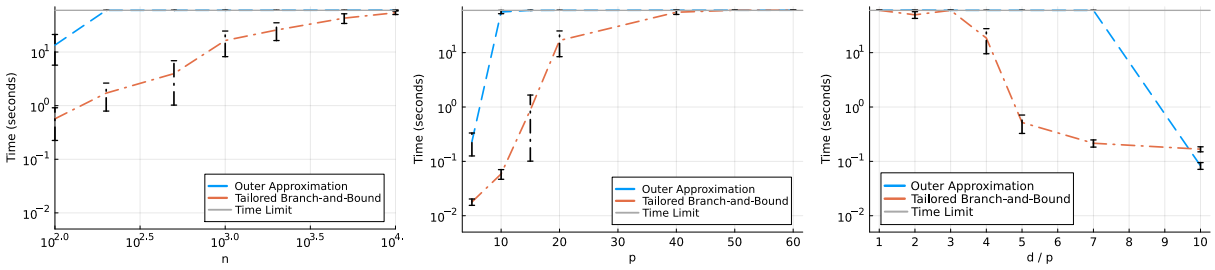
Figure 2 **Cumulative distribution of computational time (in seconds) and final gap after reaching the time limit (60 seconds) for the best implementation of the outer approximation scheme and our tailored branch-and-bound algorithm on 210 OED instances.**

terminating at the time limit, the optimality gaps achieved by our branch-and-bound are significantly lower than those of outer approximation. Overall, we can conclude that our branch-and-bound algorithm dominates outer approximation (in a stochastic dominance sense).

We further investigate the computational time of both algorithms as a function of n (fixing $p = 20$ and $d/p = 4$), p (fixing $n = 1000$ and $d/p = 4$), and d (fixing $n = 1000$ and $p = 20$). We summarize the results in Figure 3. Our first observation is that our tailored branch-and-bound algorithm scales much better with respect to n and p (Figure 3a–3b): Regarding n, it terminates significantly before the time limit even for the largest instances (in the order of 10–30 seconds $n \geq 1000$) when outer approximation does not terminate within 60 seconds for instances as small as $n = 200$. The number of dimensions p plays an important role in terms of the problem's difficulty. When $p = 5$ (Figure 3b, fixing $n = 1000$ and $d/p = 4$), outer approximation terminates within seconds but systematically hits the time limit for $p \geq 10$. Our branch-and-bound algorithm, on the other hand, scales more favorably and terminates within the time limit for p up to 20. Finally, we find that the complexity of the problem decreases as the d/p ratio increases. Unlike the relatively smooth impact of n, we observe a sharp drop in computational time around $d/p \approx 4$–5 for our method, and $d/p \approx 7$ for outer-approximation.

5. Formulations and Algorithms for OED with Routing Constraints

In this section, we consider a variant of the classical OED problem, where the sampled observations correspond to locations visited along a trajectory. Formally, instead of (1), we consider an OED problem of the form (2) where the set $\mathcal{Z} \subseteq \{0,1\}^n$ captures the trajectory constraints.



(a) Varying n (b) Varying p (c) Varying d/p

Figure 3 **Average computational time of both discrete optimization approaches when varying n (left), p (middle), and d/p ratio (right). Results are averaged over 10 simulations.**

5.1. Expressing routing constraints

We need to introduce additional constraints to ensure that the set of visited locations $\{i : z_i = 1\}$ forms an admissible trajectory. Formally, we consider an directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where the set of nodes $\mathcal{N} = \{0, 1, \ldots, n\}$ corresponds to all possible locations and $\mathcal{A} \subseteq \mathcal{N} \times \mathcal{N}$ is the set of existing arcs.[1] The location 0 corresponds to the depot or initial location of our system. For a subset of nodes $\mathcal{S} \subseteq \mathcal{N}$, we denote $\mathcal{A}(\mathcal{S})$ the set of arcs that connect two nodes in $\mathcal{S}$, and $\delta^+(\mathcal{S})$ (resp. $\delta^-(\mathcal{S})$) the set of arcs going out of (resp. in) $\mathcal{S}$, i.e., $\delta^+(\mathcal{S}) = \{(i, j) \in \mathcal{A} : i \in \mathcal{S}, j \notin \mathcal{S}\}$. For concision, we denote $\delta^+(i) := \delta^+(\{i\})$.

In our setting, an admissible trajectory corresponds to a sequence d of distinct locations that starts and ends at the depot. This constraint relates to the Orienteering Problem (OP). OP aims to find a route that visits a subset of the nodes in a graph and maximizes profit, while satisfying a cost constraint (see, e.g., Feillet et al. 2005, Vansteenwegen and Gunawan 2019). We now review different formulations for OP.

Formulation with subtour elimination constraints We introduce an auxiliary variable $\boldsymbol{y} \in \{0, 1\}^{\mathcal{A}}$ indicating the arcs that constitute the route. With these notations, $\boldsymbol{z} \in \{0, 1\}^n$ is an admissible route if there exists $\boldsymbol{y} \in \{0, 1\}^{\mathcal{A}}$ satisfying

$$\begin{cases} 1 = \sum_{a \in \delta^+(0)} y_a = \sum_{a \in \delta^-(0)} y_a, & \\ z_i = \sum_{a \in \delta^+(i)} y_a = \sum_{a \in \delta^-(i)} y_a, & \forall i \in [n], \\ \sum_{a \in \mathcal{A}} \ell_a y_a \leq d, & \\ \sum_{a \in \mathcal{A}(\mathcal{S})} y_a \leq |\mathcal{S}| - 1, & \forall \mathcal{S} \subseteq \mathcal{N} \setminus \{0\}. \end{cases} \tag{4}$$

The first two constraints ensure that there are two selected arcs (one to arrive and one to leave) for each visited location (i.e., the depot and any location i with $z_i = 1$). The second constraint is a length

(or cost) constraint. In our numerical experiments, we only impose a limit on the number of visited locations (d, including the depot), i.e., $\ell_a = 1, \forall a$. The fourth group of constraints corresponds to the subtour elimination constraints in the Dantzig–Fulkerson–Johnson formulation of the TSP (DFJ; Dantzig et al. 1954), applied to OP as in Vansteenwegen and Gunawan (2019). For every subset of nodes that does not include the depot, it forbids the presence of a cycle, hence their name. Alternatively, we can express them with outgoing arcs $\delta^+(\mathcal{S})$ instead of internal arcs. However, unlike in the TSP, we do not visit all the nodes in an OP. Because of this distinction, the subtour elimination constraints with outgoing arcs are written as follows (Fischetti and Toth 1988):

$$\sum_{a \in \delta^+(\mathcal{S})} y_a \geq z_i \quad \forall \mathcal{S} \subseteq \mathcal{N}, i \notin \mathcal{S}.$$

Observe that there is an exponential number of subtour elimination constraints, and efficient implementation relies on introducing them iteratively.

Compact formulation For the TSP, Miller et al. (1960, MTZ) propose a compact formulation involving the ordering of the visited nodes. A similar formulation exists for OP. Formally, for every node $i \in \mathcal{N}$, the integer variable $u_i \in \{0 \dots n\}$ indicates that location i is the u_ith location to be visited. We set $u_0 = 0$. Then, $\boldsymbol{z} \in \{0,1\}^n$ is an admissible route if there exists $\boldsymbol{y} \in \{0,1\}^{\mathcal{A}}$, $\boldsymbol{u} \in \{0, \dots, n\}^{\mathcal{N}}$ satisfying

$$\begin{cases} 1 = \sum_{a \in \delta^+(0)} y_a = \sum_{a \in \delta^-(0)} y_a, & \\ z_i = \sum_{a \in \delta^+(i)} y_a = \sum_{a \in \delta^-(i)} y_a, & \forall i \in [n], \\ \sum_{a \in \mathcal{A}} \ell_a y_a \leq d, & \\ u_i - u_j + 1 \leq d(1 - y_{(i,j)}), & \forall (i,j) \in \mathcal{A},\ j \neq 0. \end{cases} \tag{5}$$

Indeed, the last constraint ensures that for any sequence of locations $i_1 \to \cdots \to i_k$ that does not contain 0 (i.e., the depot), the sequence $u_{i_1}, \dots, u_{i_k}$ is strictly increasing. Consequently, it forbids the presence of any cycle that does not contain the depot. Compared to the subtour elimination formulation, this compact formulation requires $\mathcal{N}$ additional integer variables but replaces an exponential (in $|\mathcal{N}|$) number of constraints by $|\mathcal{A}| = O(|\mathcal{N}|^2)$ linear constraints. Finally, in our implementation, we use an alternative formulation of the monotonicity constraints, which is known to lead to tighter relaxations (Desrochers and Laporte 1991).

For the rest of this section, we denote

$$\mathcal{Z}_{DFJ} = \left\{ \boldsymbol{z} \in \{0,1\}^n : \exists \boldsymbol{y} \in \{0,1\}^{\mathcal{A}} \text{ s.t. (4)} \right\},$$
$$\mathcal{Z}_{MTZ} = \left\{ \boldsymbol{z} \in \{0,1\}^n : \exists \boldsymbol{y} \in \{0,1\}^{\mathcal{A}}, \boldsymbol{u} \in \{0, \dots, d\}^{\mathcal{N}} \text{ s.t. (5)} \right\},$$

and relax($\mathcal{Z}_{DFJ}$) and relax($\mathcal{Z}_{MTZ}$) the polyhedra obtained by relaxing all integrality constraints (on $\boldsymbol{z}$ as well as on the lifted variables). While most discrete optimization approaches for the OP involve either one of the two formulations, an original feature of our tailored branch-and-bound scheme is that it leverages both of them, yet in different parts of the algorithm.

5.2. Benchmark: Outer approximation

The outer-approximation strategy described in Section 3.1 is a generic approach to deal with non-linear objectives. Consequently, it can readily be adapted to the case with routing constraints by introducing additional integer $\boldsymbol{y}$ in the case of $\mathcal{Z}_{DFJ}$, $\boldsymbol{y}$ and $\boldsymbol{u}$ for $\mathcal{Z}_{MTZ}$, and the corresponding constraints. For $\mathcal{Z}_{DFJ}$, the subtour elimination constraints should be introduced iteratively as lazy constraints, like the constraints approximating the objective function.

We also compare with the method of Wang et al. (2025), which corresponds to outer approximation applied to the $\mathcal{Z}_{DFJ}$ formulation and enhanced with an additional inequality derived from Benders decomposition (for details, see section 4.3 and inequality 11 in Wang et al. 2025).

5.3. Algorithm: Tailored branch-and-bound

We now describe our branch-and-bound algorithm for OED with routing constraints. Pseudo-code is presented in Algorithm A.1.

Given two sets $\mathcal{I}_0, \mathcal{I}_1 \subseteq \{1, \dots, n\}$, we define the sub-problem $\mathcal{P}(\mathcal{I}_0, \mathcal{I}_1)$

$$\mathcal{P}(\mathcal{I}_0, \mathcal{I}_1): \quad \max_{z \in \mathcal{Z}} \operatorname{logdet}(\boldsymbol{X}^\top \operatorname{diag}(\boldsymbol{z}) \boldsymbol{X}) \text{ s.t. } z_i = 0 \text{ for } i \in \mathcal{I}_0,\ z_i = 1 \text{ for } i \in \mathcal{I}_1.$$

For a given subproblem $\mathcal{P}(\mathcal{I}_0, \mathcal{I}_1)$ (or active node), our algorithm proceeds in three steps: (i) it applies a screening rule to expand the set of excluded locations $\mathcal{I}_0$; (ii) it computes an upper bound by solving the Boolean relaxation using Frank-Wolfe; (iii) it constructs a good feasible solution. We now present each step in detail.

(i) Screening rule. The screening rule developed in Section 3.3 fixes some coordinates z_i to 0 or 1, after testing that the other value can only be a sub-optimal choice. This test relies on the ability to efficiently solve (by sorting, in the case of OED) problems of the form $\max_{z \in \mathcal{Z}\ z_i = 0/1} \boldsymbol{\delta}^\top \boldsymbol{z}$ for a fixed $\boldsymbol{\delta}$. Unfortunately, in the presence of routing constraints, even simple linear optimization problems of this form can be challenging. However, the same constraints provide an opportunity to screen variables based on feasibility considerations (instead of optimality ones). Assume that k locations $i_1, \dots, i_k$ are visited (i.e., $z_{i_1} = \dots = z_{i_k} = 1$). Intuitively, we can set $z_i = 0$ for all locations i are that 'too far' from $i_1, \dots, i_k$. Mathematically, for any location i that is not fixed by our algorithm,

we solve a small TSP problem to find the shortest cycle visiting 0 (the depot), $i_1, \ldots, i_k$, and i. If the length of this route exceeds d, we can safely fix z_i to 0. We use the DFJ formulation for these TSPs, generating subtour elimination constraints on the fly. When solving these TSPs for multiple locations i, we use the subtour elimination constraints generated at one location as warmstarts for the TSP at the next location. Details can be found in Appendix A.1.

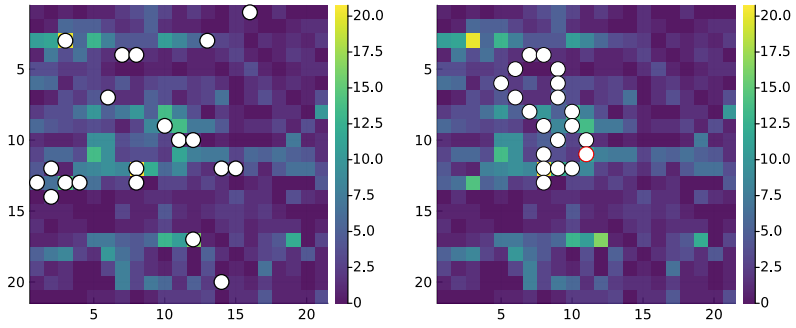
(ii) Upper bound: Relaxation solved using Frank-Wolfe. Frank-Wolfe algorithm, which is the most scalable method for the relaxation of the OED problem, can easily be adapted to the routing case. At each iteration, FW solves a linear optimization problem over $z \in \text{relax}(\mathcal{Z})$. Although no longer solvable via sorting, these optimization problems can still be solved efficiently by existing solvers. Based on preliminary experiments, we decided to use $\text{relax}(\mathcal{Z}_{MTZ})$ when solving the relaxation, because of its compactness.

(iii) Lower bound: Greedy rounding. We generalize the greedy rounding procedure, which appears to be the most efficient strategy for generating feasible solutions to OED. In essence, we want to solve —or at least find a good solution to— the optimization problem $\max_{z \in \mathcal{Z}} z^\top \hat{z}$ where $\hat{z}$ is the solution obtained from the relaxation. To do so efficiently, we reuse information obtained from the screening rule described above. Namely, given a subproblem where we fix k locations $i_1, \ldots, i_k$ to be visited, we evaluated whether we can set $z_i = 0$ for all unfixed location i, by solving a TSP on a reduced graph. We pick the location i that has not been fixed to 0 (meaning there exists a tour of length less than d visiting $i_1, \ldots, i_k$ and i in the surrogate TSP problem) and that maximizes $\hat{z}_i$. We then check whether the tour (i.e., ordered sequence of visits) provided by the surrogate TSP leads to a feasible solution for our problem (see Appendix A.1).

We should emphasize a salient feature of our algorithm: It uses both the DFJ and the MTZ formulations (for the screening rule and Boolean relaxation, respectively) unlike most existing approaches that involve a single formulation. For the relaxation, we leverage the compactness and polynomial number of constraints of the MTZ formulation, while, for the screening rule, the use of the DFJ formulation allows us to to reuse the generated subtour elimination constraints across locations i.

6. Case Study: Ensembling Ocean Plastic Density Model

We illustrate our solution to the OED problem with routing constraints on the application described in Section 1.1.



(a) Classic OED (b) OED with routing

Figure 4 **Example of the optimal data collection solution of classic OED (left panel) and OED with routing constraint (right panel), given six plastic prediction models (day 60, maps centered at location (150, 225), $d = 20$). The white circles represent the optimal solution for visiting $d = 20$ locations. The background color of each subfigure represents the predicted plastic density for the first (out of six) prediction model.**

6.1. Data description

Our data consists of 6 plastic prediction models ($p = 6$). Each model provides a density map for the entire GPGP, for each day in the year 2018. A map corresponds to a discretization of the GPGP via a 8km×8km grid, leading to $250 \times 500 = 125,000$ possible cells or locations. Formally, each day in the data can be represented by a matrix $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ where each row $i \in \{1, \ldots, n\}$ contains the $p = 6$ different predictions of plastic density made for that specific location i.

We apply our algorithms to the graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is the set of locations/cells on a map, and the arcs in $\mathcal{A}$ connect each cell in $\mathcal{N}$ with its 8 adjacent ones. We set the length of all arcs, ℓ_a, to one, so that the length constraint can be directly interpreted as a constraint on the number of locations visited. For illustration purposes, Figure 4 represents one instance from our dataset and compares the solution of the classical OED problem (Figure 4a) with that of OED with routing constraints (Figure 4b).

We consider 24 dates across our 2018 data (day 15, day 30, ..., day 360). Actually, many locations on the edge of the GPGP have zero density according to all models so we restrict our attention to the central part of the GPGP. Formally, for each date, we compute the density-weighted centers (or center of mass) of each map, and place the depot at the average of the 6 centers (rounded to the closet cell). For each day, we vary $d \in \{8, 10, 12, 15, 17, 20, 22, 25, 30, 40, 50\}$. For a given route length d, the system can only reach locations that are at a distance at most $\lceil d/2 \rceil$ away from the depot location; hence, each instance consists of six $\sqrt{n} \times \sqrt{n}$ maps (one for each of the six models) with

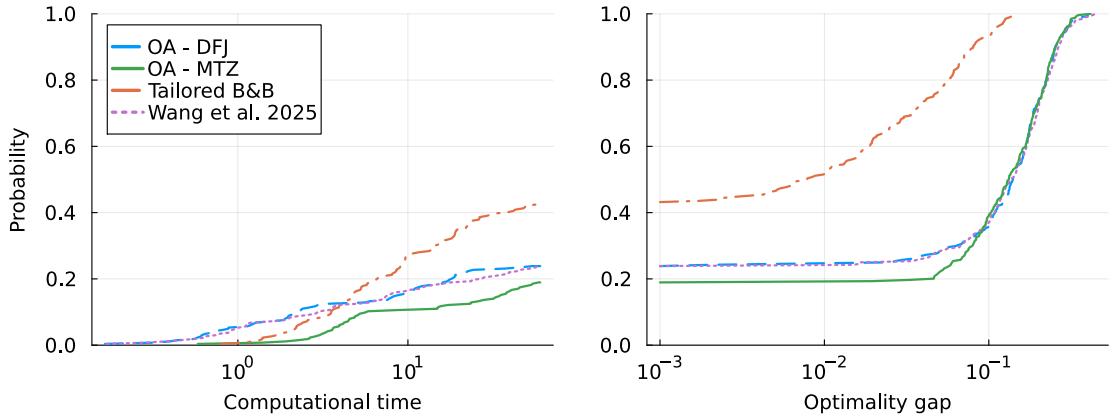


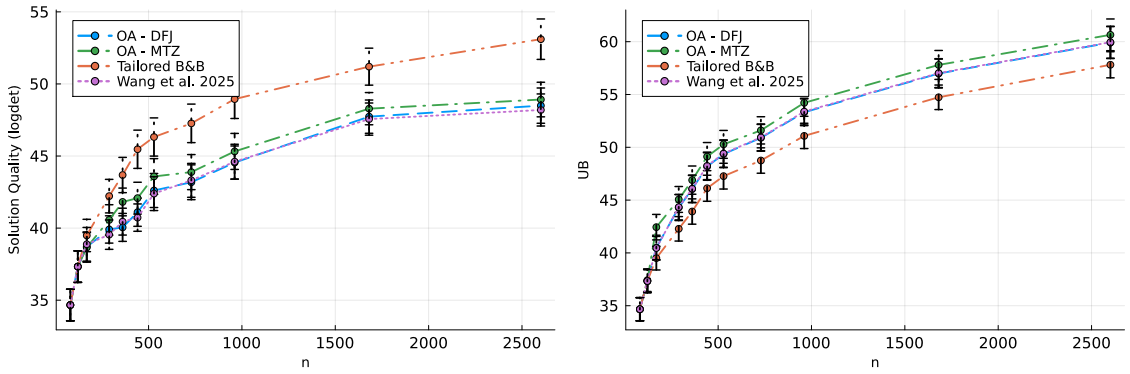
Figure 5 **Cumulative distribution of computational time (in seconds) and final gap after reaching the time limit (60 seconds) for outer approximation schemes and our tailored branch-and-bound algorithm on 264 OED instances with routing constraints.**

$\sqrt{n} = 2\lceil d/2 \rceil + 1$. Altogether, we generate $24 \times 11 = 264$ instances. Experiments are implemented in Julia and run on an Intel i7-11700K 3.60GHz CPU with 64GB of memory.

6.2. Algorithmic efficiency

On our testbed of instances, we compare the performance of our branch-and-bound algorithm ('Tailored B&B') with that of an outer-approximation scheme ('OA') implemented using the commercial solver Gurobi 11.0—as for OED without routing constraints, we warmstart OA with one constraint generated at the relaxed solution and one feasible solution from our heuristic. For the routing constraints, we consider both the DFJ and MTZ formulations. We also implement algorithm 1 of Wang et al. (2025) with their two inequalities, 10 and 11. We impose a 60-second time limit and 0.1% target optimality gap to all methods.

Figure 5 represents the cumulative distribution of time (left panel) and optimality gap at termination (right panel) for the three methods. We observe that our algorithm solves more than 40% of the instances to optimality compared with 20% for the best OA scheme (OA-DFJ). In addition, we achieve 10% optimality gap or lower on all instances, while OA achieves the same performance on less than a third of the instances. Wang et al. (2025) performs comparably to OA-DFJ. In Section EC.5.1, we report these metrics for each value of n separately. Improvement in optimality gap could come from either a better (i.e., higher) quality solution or from a better (i.e., lower bound) on the best achievable objective value. In our case, we observe in Figure 6 that both mechanisms explain the superior performance of our tailored branch-and-bound algorithm.

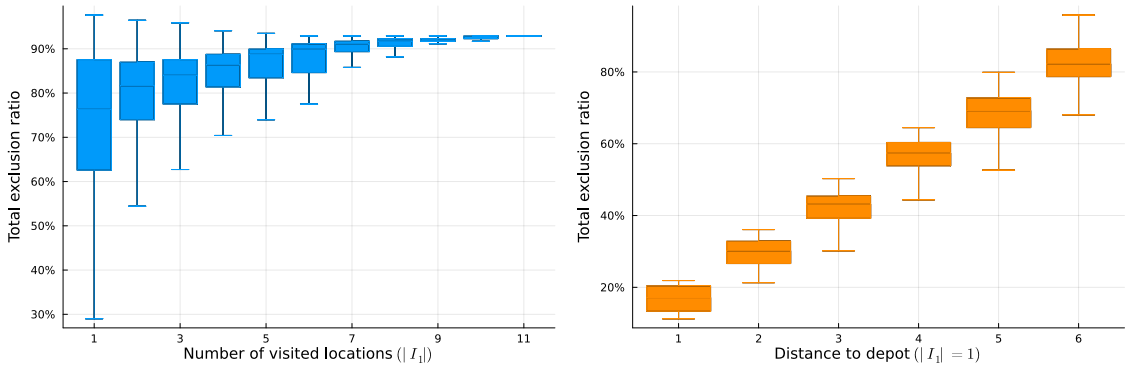


(a) Solution quality (b) Objective bound

Figure 6 **Solution quality and best bound, returned by OA schemes and our tailored branch-and-bound (B&B), on OED instances with routing constraints, as the number of locations n increases ($n = O(d^2)$).**

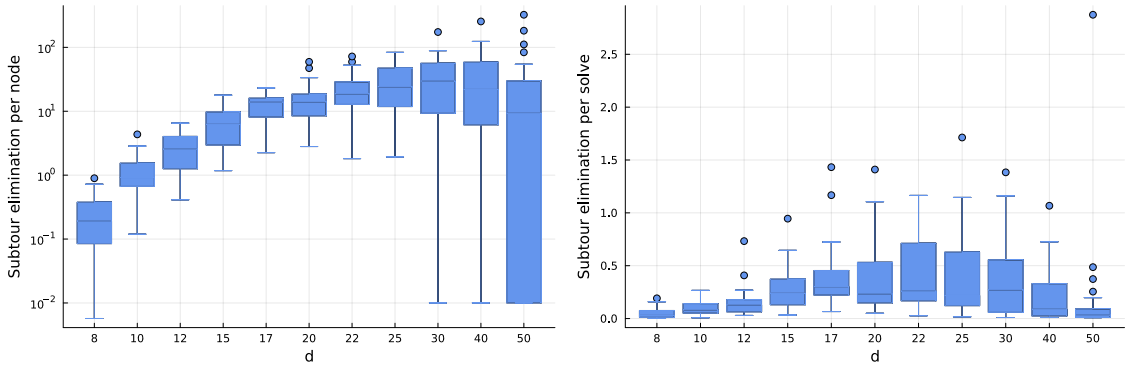
A central component of our branch-and-bound algorithm is the screening rule, namely the fact that, given a set of locations to be visited $\mathcal{I}_1$, we test whether location i is within reach given the constraint on the route length. We focus on instances with $d = 12$ (24 instances) and illustrate the effectiveness of the screening rule in Figure 7. The left panel represents the (relative) number of excluded locations, $|\mathcal{I}_0|/n$, as the number of visited locations, $|\mathcal{I}_1|$, increases. Our screening rule is extremely effective, with nearly 80% of the locations that can be safely excluded when only one visit is imposed. Intuitively, the screening rule is more effective when we are forced to visit locations that are far away from the depot (because they consume a larger fraction of the length budget). We validate this intuition in the right panel of Figure 7, where we stratify the exclusion ratio $|\mathcal{I}_0|/n$ achieved at nodes with a single visited location ($|\mathcal{I}_1| = 1$) based on the distance between the depot and that location.

To evaluate whether a location is within reach, we solve a small TSP problem in a surrogate graph (see Appendix A.1), using the subtour elimination formulation. A key driver of numerical tractability is that, for each subproblem (or node in the branch-and-bound tree), we reuse the constraints generated at one location for the other ones. Figure 8 represents the number of constraints generated per subproblem/node (left panel) and per TSP instance (right panel). We observe that the number of constraints generated per node increases moderately with the dimension of the problem d.[2] However, per TSP instance (right panel), the average number of constraints generated is largely below 1, indicating that most instances in a node can be solved without introducing new constraints. We observe similar behavior in terms of computational time (see Figure EC.9 in Section EC.5.1).



(a) As $|\mathcal{I}_1|$ increases (b) As distance to the depot increases

Figure 7 **Fraction of all excluded locations after application of the screening rule, for OED problems with routing constraint. Results are aggregated over the 24 instances with $d = 12$.**



(a) Per subproblem/node (b) Per TSP instance

Figure 8 **Distribution (boxplot) of the number of subtour elimination constraints generated as the problem dimension d increases.**

We also investigate in Section EC.5.1 how our heuristic for finding high-quality solutions benefits from branching.

6.3. Impact on downstream prediction

As mentioned in Section 1.2, our motivation for solving the OED problem with routing constraints (2) is to collect data to build a model ensemble. In this section, we showcase the benefit of our algorithm on downstream predictive accuracy.

To do so, we generate a true plastic density y according to the linear model $y = x_{j^\star} + \epsilon$, where $j^\star \in \{1, \ldots, 6\}$ and $\epsilon \sim \mathcal{N}(0, \sigma^2)$ is a noise term that we take i.i.d. across locations. We calibrate σ^2 so that the signal-to-noise ratio $\text{var}(x_{j^\star})/\sigma^2$ is equal to 0.5.[3] In other words, we consider a setting where the truth is a noisy version of one of the six models. In this setting, we simulate a

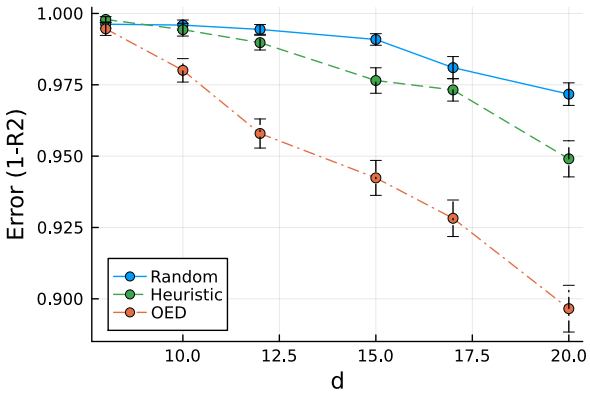
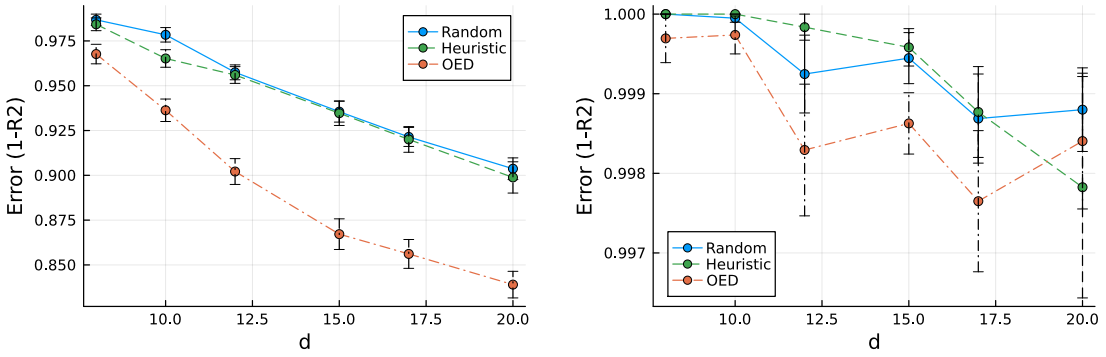


Figure 9 **Average error ($1 - R^2$) of the downstream regressor depending on the method used for collecting data: random, a linearization heuristic (STD, see Section EC.3.1), or by optimal experimental design (OED). For each value of d, results are averaged over 24 instances and 20 generations of ground truth data.**

situation where the ground truth data is collected at d locations, a linear model $\hat{\boldsymbol{\beta}}$ is estimated from this data, and predictions $\boldsymbol{x}^\top \hat{\boldsymbol{\beta}}$ are made for the entire region. In Figure 9, we report the normalized squared error ($1 - R^2$) achieved depending on the method used to decide which locations to visit, between a random design (obtained by solving $\max_{z \in \mathcal{Z}} \boldsymbol{r}^\top \boldsymbol{z}$ for some $\boldsymbol{r} \sim \mathcal{N}(0, \boldsymbol{I}_n)$), the solution of a linearization heuristic (STD; see Section EC.3.1), and our OED with routing constraints algorithm. Note that our metric aggregates errors made in- and out-of-sample. However, since $d \ll n$, it primarily corresponds to out-of-sample error. For a fixed budget d, we observe that choosing an optimal design significantly reduces the prediction error. Alternatively, for a target prediction error, our OED solution reduces the number of required locations (i.e., the cost of data acquisition) by 30 to 50% compared with the STD heuristic or a random design strategy, respectively.

Of course, the current evaluation methodology corresponds to a well-specified case where the generating model for y satisfies all the assumptions underpinning the OED formulation (1). As a robustness check, we now consider two alternative models, which violate the i.i.d noise assumption and both the linearity and i.i.d. assumptions respectively. First, we consider a model where the error terms $\boldsymbol{\epsilon}$ are sampled as a scaled version of one of the maps x_j's (for some randomly sampled j). Second, we create non-linearity by generating y as $y = x^s_{j^\star} + \epsilon$ where $x^s_{j^\star}$ is a shifted version of the map provided by model $j^\star$ (and we use a random longitude/latitude shift of at most 3 locations). As displayed in Figure 10, we observe that the OED solution still provides noticeable benefits compared with the other data collection strategies, even under these misspecifications. Note that, as reviewed in Section 1.2, we could have adapted our optimization formulation to account for these misspecifications, e.g., by including shifted versions of each map j as additional dimensions of $\boldsymbol{x}_i$



(a) Non i.i.d. noise (b) Non-linear + non i.i.d. noise

Figure 10 **Average error $(1-R^2)$ of the downstream regressor for misspecified model depending on the method used for collecting data: random, a linearization heuristic (STD, see Section EC.3.1), or our tailored branch-and-bound. For each value of d, results are averaged over 24 instances and 20 generations of ground true data.**

(hence, increasing the dimension p). Doing so would have further improved the performance of our approach.

Instead of prediction error, one could be interested in the ability to identify the best model (i.e., the accuracy in identifying $j^\star$). We report results for this metric instead in Section EC.5.2.

6.4. Modeling flexibility

One benefit of formal discrete optimization methods is the additional modeling flexibility. Instead of optimizing only for an information-related objective, our algorithms can readily be applied to problems of the form

$$\max_{z \in \mathbf{Z}} \quad \lambda \cdot \boldsymbol{r}^\top \boldsymbol{z} + (1-\lambda) \cdot \operatorname{logdet}(\boldsymbol{X}^\top \operatorname{diag}(\boldsymbol{z}) \boldsymbol{X}) \quad \text{s.t.} \quad \sum_{i=1}^{n} z_i \leq d, \tag{6}$$

where $\boldsymbol{r} \in \mathbb{R}^n$ is a given reward vector and $\lambda \in [0,1]$ a tuning parameter trading-off between the two objectives. For example, in our use case, The NGO is also interested in collecting data primarily from regions with a higher density of plastic (e.g., to allow for the joint deployment of data-collecting and plastic-collecting devices). To do so, they could be interested in a problem of the form (6) with a reward r_i corresponding to an estimate of plastic density in location i (such as the average of all model predictions $\boldsymbol{r} = \frac{1}{p}\boldsymbol{X}\boldsymbol{e}$). Through the parameter λ, they control the importance of visiting a high-density region compared to a high-information one.

Figure 11 visualizes the trade-off between plastic density encountered ($\boldsymbol{r}^\top \boldsymbol{z}$) and prediction error $(1-R^2)$ for 24 instances (and 10,000 samples of ground truth data y) with $d = 15$. We observe that,

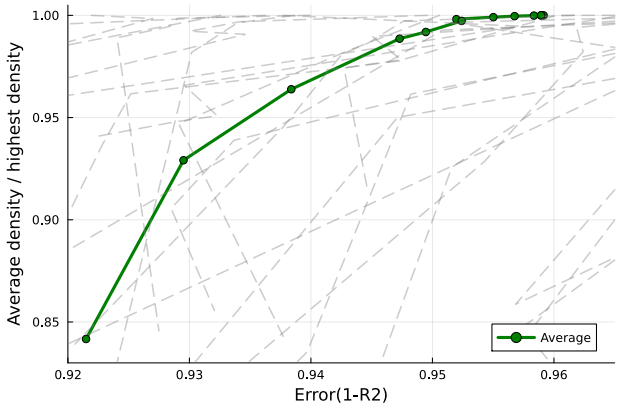


Figure 11 **Pareto plot of prediction error (x-axis, $1 - R^2$) and relative plastic density encountered (y-axis, $\boldsymbol{r}^\top \boldsymbol{z}$, standardized by highest value of each instance) as λ varies. Dashed grey lines report the results over each individual instances (24 in total). The solid green line report the results averaged across instances.**

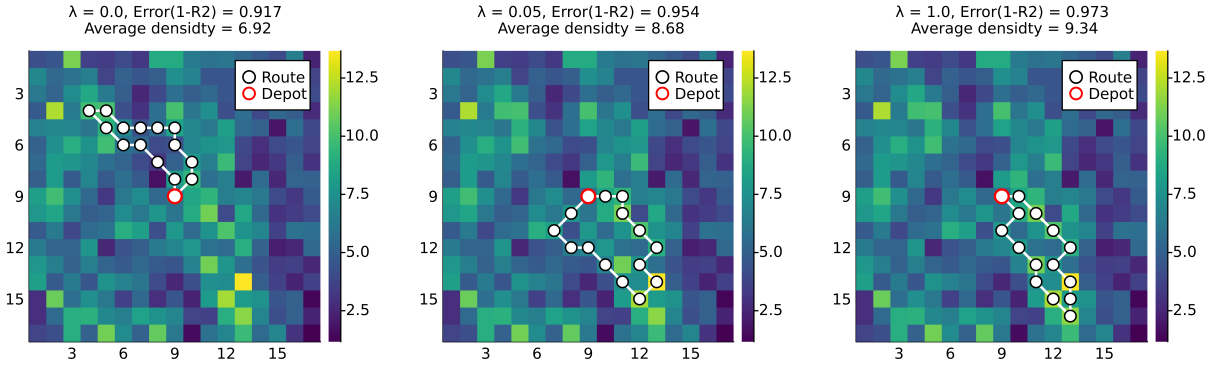


Figure 12 **Example of trajectories (visiting $d = 15$ locations) obtained via solving (6) for different λ values. The background color of each subfigure represents the average predicted plastic density of the six prediction models.**

for $\lambda = 0$ (bottom left of the curve), the optimal route achieves the lowest error (92%) but encounters only 84% of the maximum plastic density (100% corresponds to the solution for $\lambda = 1$). By slightly increasing λ, we can find routes that perform marginally worse in terms of prediction error (from 92% to 93%) but encounter a much higher density of plastic (from 84% to 93%). Operationally, increasing λ shifts the trajectory from high-information to higher-density regions. For example, on one instance displayed in Figure 12, the trajectory shifts from exploring the north east to the south west as λ increases.

7. Conclusion

In this paper, we study the D-optimal experimental design problem with orienteering constraints, in order to find optimal data collection strategies to construct ensemble models for ocean plastic density prediction.

First, we revisit the solution to the classical D-optimal experimental design problem. In addition to a new screening rule, our algorithm relies on an efficient first-order algorithm (Frank-Wolfe algorithm to be precise), which solves the Boolean relaxation without any approximation of the log-determinant objective function at each node and strongly benefits from warmstarting. Numerically, it outperforms commercial branch-and-bound codes significantly, e.g., increasing the size of instances that can be solved under a minute from $n = 200$ to $n = 10,000$.

We then extend this algorithmic strategy in presence of routing constraints inherited from the orienteering problem. Indeed, many applications like ours require data collection over large and often remote areas, using unmanned vehicles such as drones. Compared with alternative solutions, a salient feature of our algorithm is that it invokes two competitive formulations for the orienteering constraints, leveraging the respective merit of each formulation at different point in the branch-and-bound process. Evaluated on real data from our partner NGO, our original algorithm outperforms existing approaches, in terms of solution quality, optimality gap at termination, and computational time. Furthermore, we investigate the effect of optimally collecting data on accuracy of the downstream predictive model, and find that our optimal design can achieve the same accuracy as alternative data collection strategy at a much lower cost.

Overall, our study provides new and numerically efficient algorithms for the well-studied OED problem from statistics, while integrating practically relevant routing constraints. Future research could further refine the modeling and the algorithm to incorporate other aspects that are meaningful in practice, or explore other examples of data science with operational constraints.

Notes

[1] Any undirected graph can be converted into a symmetric directed one by duplicating all the edges. The DFJ formulation, however, can be written in a more compact form by working with the undirected graph directly.

[2] When d is large, the number of constraints per node decreases. This is an artifact of the total time limit (see Figure EC.9), which prevents all the necessary TSP instances from being solved.

[3] A signal-to-noise ratio of 0.5 corresponds to a regression model with $R^2 = 0.33$. We calibrated this value using the fact that the ratio $\text{var}(\boldsymbol{x}_{j_1})/\text{var}(\boldsymbol{x}_{j_2} - \boldsymbol{x}_{j_1})$ is equal to 0.79 on average in our data. Hence, 0.5 appeared as a conservative realistic estimate.

Appendix A: Additional Description of Branch-and-Bound Algorithm

A.1. Surrogate TSP problem for coordinate screening

Given a set of locations $\mathcal{V} := \{0, i_1, \ldots, i_k, i\}$, we construct a fully connected directed graph $\tilde{\mathcal{G}} = (\mathcal{V}, \mathcal{V} \times \mathcal{V})$ and assign to each arc $a = (i, j) \in \mathcal{V} \times \mathcal{V}$ a length, corresponding to the shortest path from i to j in the original graph $\mathcal{G}$, i.e.,

$$
\begin{aligned}
w_{(i,j)} := \min_{\boldsymbol{y} \in \mathbb{R}_+^{\mathcal{A}}} \sum_{e \in \mathcal{A}} \ell_a y_a \quad \text{s.t.} \quad & \sum_{a \in \delta^+(i)} y_a = 1, \\
& \sum_{a \in \delta^-(j)} y_a = 1, \\
& \sum_{a \in \delta^+(v)} y_a = \sum_{a \in \delta^-(v)} y_a, \quad \forall v \in \mathcal{V}.
\end{aligned} \tag{7}
$$

Note that this is a classical network flow problem, and its solution provides the shortest path (in the original graph $\mathcal{G}$) between i and j. In addition, if the set of nodes is equipped with some metric structure and the original graph is symmetric, w_a can often be computed directly as the distance between i and j (e.g., the Manhattan distance in a grid-like network).

We then solve a small TSP for finding the minimum-length cycle visiting all the nodes $\mathcal{V}$ in $\tilde{\mathcal{G}}$:

$$
\begin{aligned}
\min_{\boldsymbol{y} \{0,1\}^{\mathcal{V} \times \mathcal{V}}} \sum_{e \in \mathcal{V} \times \mathcal{V}} w_a y_a \quad \text{s.t.} \quad & \sum_{a \in \delta^+(i)} y_a = \sum_{a \in \delta^-(i)} y_a = 1, \ \forall i \in \mathcal{V}, \\
& \sum_{a \in \mathcal{A}(\mathcal{S})} y_a \leq |\mathcal{S}| - 1, \ \forall \mathcal{S} \subseteq \mathcal{V} \setminus \{i\}.
\end{aligned}
$$

If the value of this TSP exceeds d, we can fix $z_i = 0$ in the rest of our algorithm. The subtour elimination constraints are introduced dynamically using lazy constraints. In addition, because we are solving similar TSP instances for different sets of nodes $\mathcal{V}$ that only differ by one location i, we can reuse the cuts generated for one instance to the next one.

Finally, the solution to this TSP problem in $\tilde{\mathcal{G}}$ defines an ordering of the nodes in $\mathcal{V}$, $0 = v_0 \to v_1 \to \cdots \to v_{k+1} \to v_0$. Each transition $v_j \to v_{j+1}$ corresponds to a path in $\mathcal{G}$. Hence, by concatenating these paths, we obtain a route in $\mathcal{G}$. However, this route is not guaranteed to be feasible for our original problem because the same location can be visited more than once, as part of the transition $v_j \to v_{j+1}$ and $v_{j'} \to v_{j'+1}$ for $j \neq j'$ for example. This issue, however, does not occur if the original graph $\mathcal{G}$ satisfies the following assumption: For any pair of nodes $(i, j) \in \mathcal{N} \times \mathcal{N}$, a direct arc connecting i to j exists, i.e., $(i, j) \in \mathcal{A}$, and is length minimizing, i.e., $\ell_{(i,j)} = w_{(i,j)}$ with $w_{(i,j)}$ defined by (7).

A.2. Tailored branch-and-bound code

Algorithm A.1: Tailored branch-and-bound scheme for OED with routing constraints (2)

input Target gap ϵ;

$\vdots$

Initialize $\mathcal{A} \leftarrow \{\mathcal{P}(\emptyset, \emptyset)\}$;

Initialize $(UB, LB) \leftarrow (+\infty, -\infty)$, $z^\star \leftarrow \mathbf{0}$;

while $\mathcal{A} \neq \emptyset$ *and* $UB - LB > \epsilon|LB|$ **do**

Pop `node` (associated with $\mathcal{P}(\mathcal{I}_0, \mathcal{I}_1)$) from $\mathcal{A}$ with highest upper bound ;

`// Screening rule`

for $i \notin \mathcal{I}_0 \cup \mathcal{I}_1$ **do**

Solve surrogate TSP visiting $\mathcal{I}_1 \cup \{i\}$;

If length of tour exceeds d, $\mathcal{I}_0 \leftarrow \mathcal{I}_0 \cup \{i\}$;

end

`// Upper bound`

Solve the Boolean relaxation of $\mathcal{P}(\mathcal{I}_0, \mathcal{I}_1)$ using FW (Algorithm EC.2.1), obtain $\bar{z}$;

Set `node.ub` $\leftarrow f(\bar{z})$;

`// Lower bound`

Select $i_0 \in \arg\max_i \{\bar{z}_i : i \notin \mathcal{I}_0 \cup \mathcal{I}_1\}$;

Construct feasible solution visiting $\mathcal{I}_1 \cup \{i_0\}$, z_0 ;

Set `node.lb` $\leftarrow f(z_0)$;

if *node.lb* $> LB$ **then**

Set $LB \leftarrow f(z_0)$, $z^\star \leftarrow z_0$;

end

`// Branching`

Find $i^\star \in \arg\max_i \{|\nabla f(\bar{z}_i)| : \bar{z}_i(1 - \bar{z}_i) > 0\}$;

Define `left` and `right` child nodes, associated with $\mathcal{P}(\mathcal{I}_0 \cup \{i^\star\}, \mathcal{I}_1)$ and $\mathcal{P}(\mathcal{I}_0, \mathcal{I}_1 \cup \{i^\star\})$;

Set `left.ub` = `right.ub` = `node.ub` ;

Push `left` and `right` to $\mathcal{A}$;

`// Pruning`

Filter $\mathcal{A} \leftarrow \{\mathtt{a} \in \mathcal{A} : \mathtt{a.ub} > LB\}$;

Update $UB = \max\{\mathtt{a.ub} : \mathtt{a} \in \mathcal{A}\}$;

end

References

Ahipasaoglu SD (2021) A branch-and-bound algorithm for the exact optimal experimental design problem. *Statistics and Computing* 31(5):65.

Allen-Zhu Z, Li Y, Singh A, Wang Y (2017) Near-optimal design of experiments via regret minimization. *International Conference on Machine Learning*, 126–135 (PMLR).

Anstreicher KM (2020) Efficient solution of maximum-entropy sampling problems. *Operations Research* 68(6):1826–1835.

Atkinson AC (1996) The usefulness of optimum experimental designs. *Journal of the Royal Statistical Society Series B: Statistical Methodology* 58(1):59–76.

Atkinson AC, Fedorov V (1975) The design of experiments for discriminating between two rival models. *Biometrika* 62(1):57–70.

Bach E, Ghil M (2023) A multi-model ensemble kalman filter for data assimilation and forecasting. *Journal of Advances in Modeling Earth Systems* 15(1):e2022MS003123.

Bauer P, Linderoth JT, Savelsbergh MW (2002) A branch and cut approach to the cardinality constrained circuit problem. *Mathematical Programming* 91:307–348.

Bertsimas D, Cory-Wright R, Pauphilet J (2021) A unified approach to mixed-integer optimization problems with logical constraints. *SIAM Journal on Optimization* 31(3):2340–2367.

Bertsimas D, Cory-Wright R, Pauphilet J (2022) Solving large-scale sparse PCA to certifiable (near) optimality. *Journal of Machine Learning Research* 23(13):1–35.

Bertsimas D, Lamperski J, Pauphilet J (2020a) Certifiably optimal sparse inverse covariance estimation. *Mathematical Programming* 184(1):491–530.

Bertsimas D, Pauphilet J, Van Parys B (2020b) Sparse regression: Scalable algorithms and empirical performance. *Statistical Science* 35(4):555–578.

Bischoff W (1992) On exact D-optimal designs for regression models with correlated observations. *Annals of the Institute of Statistical Mathematics* 44:229–238.

Bouhtou M, Gaubert S, Sagnol G (2010) Submodularity and randomized rounding techniques for optimal experimental design. *Electronic Notes in Discrete Mathematics* 36:679–686.

Box GE, Draper NR (1959) A basis for the selection of a response surface design. *Journal of the American Statistical Association* 54(287):622–654.

Chernoff H (1953) Locally optimal designs for estimating parameters. *The Annals of Mathematical Statistics* 586–602.

Dantzig G, Fulkerson R, Johnson S (1954) Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America* 2(4):393–410.

Desrochers M, Laporte G (1991) Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Operations Research Letters* 10(1):27–36.

Dette H, Pepelyshev A, Zhigljavsky A (2015) Optimal designs in regression with correlated errors. *The Annals of Statistics* 44(1):113.

DuMouchel W, Jones B (1994) A simple bayesian modification of D-optimal designs to reduce dependence on an assumed model. *Technometrics* 36(1):37–47.

Duran MA, Grossmann IE (1986) An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming* 36:307–339.

Fedorov VV (2010) Optimal experimental design. *Wiley Interdisciplinary Reviews: Computational Statistics* 2(5):581–589.

Fedorov VV, Leonov SL (2013) *Optimal Design for Nonlinear Response Models* (CRC Press).

Feillet D, Dejax P, Gendreau M (2005) The profitable arc tour problem: Solution with a branch-and-price algorithm. *Transportation Science* 39(4):539–552.

Fischetti M, Toth P (1988) An additive approach for the optimal solution of the prize collecting traveling salesman problem. *Vehicle Routing: Methods and studies* 231:319–343.

Gendreau M, Laporte G, Semet F (1998) A branch-and-cut algorithm for the undirected selective traveling salesman problem. *Networks: An International Journal* 32(4):263–273.

Goos P, Kobilinsky A, O'Brien TE, Vandebroek M (2005) Model-robust and model-sensitive designs. *Computational Statistics & Data Analysis* 49(1):201–216.

Gunawan A, Lau HC, Vansteenwegen P (2016) Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research* 255(2):315–332.

Hazimeh H, Mazumder R, Saab A (2022) Sparse regression at scale: Branch-and-bound rooted in first-order optimization. *Mathematical Programming* 196(1):347–388.

Hoel PG (1958) Efficiency problems in polynomial estimation. *The Annals of Mathematical Statistics* 1134–1145.

Joshi S, Boyd S (2008) Sensor selection via convex optimization. *IEEE Transactions on Signal Processing* 57(2):451–462.

Kaandorp ML, Lobelle D, Kehl C, Dijkstra HA, Van Sebille E (2023) Global mass of buoyant marine plastics dominated by large long-lived debris. *Nature Geoscience* 16(8):689–694.

Keller CP (1989) Algorithms to solve the orienteering problem: A comparison. *European Journal of Operational Research* 41(2):224–231.

Kharin VV, Zwiers FW (2002) Climate predictions with multimodel ensembles. *Journal of climate* 15(7):793–799.

Kiefer J (1959) Optimum experimental designs. *Journal of the Royal Statistical Society: Series B (Methodological)* 21(2):272–304.

Ko CW, Lee J, Wayne K (1994) A spectral bound for D-optimality. *MODA* .

Krishnamurti T, Kishtawal CM, LaRow TE, Bachiochi DR, Zhang Z, Williford CE, Gadgil S, Surendran S (1999) Improved weather and seasonal climate forecasts from multimodel superensemble. *Science* 285(5433):1548–1550.

Krishnamurti T, Kumar V, Simon A, Bhardwaj A, Ghosh T, Ross R (2016) A review of multimodel superensemble forecasting for weather, seasonal climate, and hurricanes. *Reviews of Geophysics* 54(2):336–377.

Laporte G, Martello S (1990) The selective travelling salesman problem. *Discrete Applied Mathematics* 26(2-3):193–207.

Lau LC, Zhou H (2020) A spectral approach to network design. *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, 826–839.

Lau LC, Zhou H (2022) A local search framework for experimental design. *SIAM Journal on Computing* 51(4):900–951.

Leifer AC, Rosenwein MB (1994) Strong linear programming relaxations for the orienteering problem. *European Journal of Operational Research* 73(3):517–523.

Li Y, Fampa M, Lee J, Qiu F, Xie W, Yao R (2024) D-optimal data fusion: Exact and approximation algorithms. *INFORMS Journal on Computing* 36(1):97–120.

Li Y, Xie W (2024) Best principal submatrix selection for the maximum entropy sampling problem: scalable algorithms and performance guarantees. *Operations Research* 72(2):493–513.

López-Fidalgo J (2023) Optimal experimental design. *Lecture Notes in Statistics* 226.

Madan V, Singh M, Tantipongpipat U, Xie W (2019) Combinatorial algorithms for optimal design. *Conference on Learning Theory*, 2210–2258 (PMLR).

Miller CE, Tucker AW, Zemlin RA (1960) Integer programming formulation of traveling salesman problems. *Journal of the ACM* 7(4):326–329.

Nguyen NK, Miller AJ (1992) A review of some exchange algorithms for constructing discrete D-optimal designs. *Computational Statistics & Data Analysis* 14(4):489–498.

Nikolov A, Singh M, Tantipongpipat U (2022) Proportional volume sampling and approximation algorithms for A-optimal design. *Mathematics of Operations Research* 47(2):847–877.

Ponte G, Fampa M, Lee J (2023) Branch-and-bound for D-optimality with fast local search and variable-bound tightening. *arXiv preprint arXiv:2302.07386* .

Pukelsheim F (2006) *Optimal Design of Experiments* (SIAM).

Sagnol G, Harman R (2015) Computing exact D-optimal designs by mixed integer second-order cone programming. *The Annals of Statistics* 43(5):2198–2224.

Singh M, Xie W (2020) Approximation algorithms for D-optimal design. *Mathematics of Operations Research* 45(4):1512–1534.

Vansteenwegen P, Gunawan A (2019) Orienteering problems. *EURO Advanced Tutorials on Operational Research* 1.

Waite TW, Woods DC (2022) Minimax efficient random experimental design strategies with application to model-robust design for prediction. *Journal of the American Statistical Association* 117(539):1452–1465.

Wang J, Xie W, Ryzhov IO (2024) Algorithms for budget-constrained d-optimal design .

Wang J, Xie W, Ryzhov IO, Marković N, Ou G (2025) D-optimal orienteering for post-earthquake reconnaissance planning. *Operations Research* .

Wang Y, Yu AW, Singh A (2017) On computationally tractable selection of experiments in measurement-constrained regression models. *The Journal of Machine Learning Research* 18(1):5238–5278.

Welch WJ (1982a) Algorithmic complexity: three NP-hard problems in computational statistics. *Journal of Statistical Computation and Simulation* 15(1):17–25.

Welch WJ (1982b) Branch-and-bound search for experimental designs based on D optimality and other criteria. *Technometrics* 24(1):41–48.

Wong WK (1994) Comparing robust properties of A, D, E and G-optimal designs. *Computational Statistics & Data Analysis* 18(4):441–448.

Wu CF (1981) On the robustness and efficiency of some randomized designs. *The Annals of Statistics* 1168–1177.

Yang M, Biedermann S, Tang E (2013) On optimal designs for nonlinear models: a general and efficient algorithm. *Journal of the American Statistical Association* 108(504):1411–1420.

Zhao R, Freund RM (2023) Analysis of the frank–wolfe method for convex composite optimization involving a logarithmically-homogeneous barrier. *Mathematical programming* 199(1):123–163.

Electronic Companion: Operationalizing Experimental Design

Appendix EC.1: Preliminary Results on the D-Optimal Design Objective

The log-determinant objective in the D-optimal design problem (1) is a concave function of the variable z and its first-order derivative can be computed efficiently. We recall these (known) results in this section.

First, we reformulate the objective function in (1) as a minimization problem, by leveraging strong duality.

Lemma EC.1. *For any $z \in [0,1]^n$,*

$$\operatorname{logdet}\left(\boldsymbol{X}^\top \operatorname{diag}(z)\boldsymbol{X}\right) = \inf_{\boldsymbol{A}\geq \boldsymbol{0}} \quad \sum_{i=1}^{n} \left(\boldsymbol{x}_i^\top \boldsymbol{A}\boldsymbol{x}_i\right) z_i - \operatorname{logdet}(\boldsymbol{A}) - p.$$

Furthermore, when $\boldsymbol{X}^\top \operatorname{diag}(z)\boldsymbol{X} \succ \boldsymbol{0}$, the infimum is attained at $\boldsymbol{A}^\star = \left(\boldsymbol{X}^\top \operatorname{diag}(z)\boldsymbol{X}\right)^{-1}$.

Proof of Lemma EC.1 Fix $z \in [0,1]^n$. For any closed and convex function $g : \mathcal{S}_+^n \to \mathbb{R}$, we can write $g(\boldsymbol{W}) = \sup_{\boldsymbol{A}\in\mathcal{S}^n} \langle \boldsymbol{W}, \boldsymbol{A}\rangle - g^*(\boldsymbol{A})$ where g^* is a convex function known as the Fenchel conjugate of g (see Bertsekas 2016, section 6.4). Applying this result to $g(\boldsymbol{W}) := -\operatorname{logdet}(\boldsymbol{W})$ with $\boldsymbol{W} = \boldsymbol{X}^\top \operatorname{diag}(z)\boldsymbol{X}$ yields

$$\operatorname{logdet}\left(\boldsymbol{X}^\top \operatorname{diag}(z)\boldsymbol{X}\right) \quad = -g\left(\boldsymbol{X}^\top \operatorname{diag}(z)\boldsymbol{X}\right) \quad = \inf_{\boldsymbol{A}\in\mathcal{S}^n} g^*(\boldsymbol{A}) - \langle \boldsymbol{A}, \boldsymbol{X}^\top \operatorname{diag}(z)\boldsymbol{X}\rangle.$$

The result follows after observing that

$$g^*(\boldsymbol{A}) := \sup_{\boldsymbol{W}\succ 0} \langle \boldsymbol{A}, \boldsymbol{W}\rangle + \operatorname{logdet}(\boldsymbol{W}) \quad = -p - \operatorname{logdet}(-\boldsymbol{A})$$

from the first-order conditions $\boldsymbol{A} + \boldsymbol{W}^{-1} = \boldsymbol{0}$. □

Remark EC.1. From the proof of Lemma EC.1, we see that this reformulation as a minimization problem (and our subsequent algorithmic strategy) can be generalized to any objective function of the form $-g(\boldsymbol{X}^\top \operatorname{diag}(z)\boldsymbol{X})$ with g closed and convex.

In particular, Lemma EC.1 shows that, for any $z \in [0,1]^n$,

$$\operatorname{logdet}\left(\boldsymbol{X}^\top \operatorname{diag}(z)\boldsymbol{X}\right) \leq \sum_{i=1}^{n} \left(\boldsymbol{x}_i^\top \boldsymbol{A}\boldsymbol{x}_i\right) z_i - \operatorname{logdet}(\boldsymbol{A}) - p,\ \forall \boldsymbol{A} \succ \boldsymbol{0}.$$

Furthermore, if $\boldsymbol{A}$ is chosen optimally with respect to z, the inequality above can be interpreted as the linear approximation of a concave function by its tangent, as summarized in the following lemma—a consequence of Lemma EC.1 and Danskin's theorem (proposition B.22 Bertsekas 2016).

Lemma EC.2. *The function* $f(z) := \log\det\left(\boldsymbol{X}^\top \operatorname{diag}(z)\boldsymbol{X}\right)$ *is concave over* $[0,1]^n$ *and for any* $z \in [0,1]^n$ *such that* $\boldsymbol{X}^\top \operatorname{diag}(z)\boldsymbol{X} \succ \boldsymbol{0}$,

$$\frac{\partial f(z)}{\partial z_i} = \boldsymbol{x}_i^\top \left(\boldsymbol{X}^\top \operatorname{diag}(z)\boldsymbol{X}\right)^{-1} \boldsymbol{x}_i.$$

Appendix EC.2: Boolean Relaxation: Literature Review and Numerical Evaluation

We review and evaluate different algorithms for solving the Boolean relaxation of (1). In our preliminary numerical experiments (Section EC.2.3), we find that Frank-Wolfe largely outperforms all other approaches (by 1–2 orders of magnitude) and solves all instances in less than a second. In addition, it can be efficiently warm-started, which is crucial given that we will solve such a relaxation at all nodes in the branch-and-bound tree. As a result, we use Frank-Wolfe in our implementation.

EC.2.1. Literature review

The Boolean relaxation of (1) can simply be written in primal form

$$\max_{z\in[0,1]^n \,:\, z^\top \boldsymbol{e} \le d} \quad \operatorname{logdet}\left(\boldsymbol{X}^\top \operatorname{diag}(z)\boldsymbol{X}\right), \tag{EC.1}$$

and solved via a similar outer-approximation strategy as its discrete version—outer approximation for convex *continuous* optimization is also referred to as Kelley's algorithm (Kelley 1960). The main advantage of this approach is that it generates linear constraints that are globally valid and can be reused when solving (3) Alternatively, Joshi and Boyd (2008) propose a Newton's method to solve a smoothed approximation of (EC.1). In the context of maximum entropy sampling (and subsequently Li et al. 2024, for D-optimal data fusion), Li and Xie (2024) implement a Frank-Wolfe algorithm for solving (EC.1) and prove its (sub-linear) convergence.

Alternatively, generic interior point method codes can solve the Boolean relaxation (EC.1) in its semidefinite primal form

$$\max_{\substack{z\in[0,1]^n \,:\, z^\top \boldsymbol{e} \le d \\ \boldsymbol{W} \succeq \boldsymbol{0}}} \quad \operatorname{logdet}(\boldsymbol{W}) \ \text{ s.t. } \boldsymbol{W} \preceq \boldsymbol{X}^\top \operatorname{diag}(z)\boldsymbol{X}, \tag{EC.2}$$

or its equivalent dual formulation

$$\min_{\boldsymbol{A}\succeq\boldsymbol{0},\boldsymbol{\mu},\lambda\ge 0} \quad \boldsymbol{\mu}^\top \boldsymbol{e} + \lambda d - \operatorname{logdet}(\boldsymbol{A}) - p \ \text{ s.t. } \mu_i + \lambda \ge \boldsymbol{x}_i^\top \boldsymbol{A}\boldsymbol{x}_i, \forall i = 1,\dots,n. \tag{EC.3}$$

Finally, instead of viewing the Boolean relaxation as a constrained convex and differentiable optimization problem (either in the primal or in the dual space), we can obtain a saddle-point formulation as a direct consequence of Lemma EC.1 (proof omitted):

$$\min_{\boldsymbol{A}\succeq\boldsymbol{0}} \quad q(\boldsymbol{A}), \ \text{ with } \ q(\boldsymbol{A}) := \max_{z\in[0,1]^n : \boldsymbol{e}^\top z \le d} \quad \sum_{i=1}^{n} \left(\boldsymbol{x}_i^\top \boldsymbol{A}\boldsymbol{x}_i\right) z_i - \operatorname{logdet}(\boldsymbol{A}) - p, \tag{EC.4}$$

and interpret it as the minimization of a non-differentiable convex function $q(\boldsymbol{A})$. In the context of sparse regression, Bertsimas et al. (2020) propose a sub-gradient ascent scheme to solve Boolean relaxations with similar structure, while generating high-quality binary solutions.

EC.2.2. Implementation details

Frank-Wolfe algorithm We provide the pseudo-code of the Frank-Wolfe algorithm for solving (EC.1) in Algorithm EC.2.1. The algorithm has two stopping criteria: a bound on the total number of iterations, $T_{\texttt{max}}$ (equivalently, we could impose a time limit) and a target optimality gap, $\epsilon_{\texttt{target}}$.

Algorithm EC.2.1: Frank-Wolfe algorithm for solving (EC.1)

initialize $z^0 \leftarrow \frac{d}{n}\mathbf{1}$;
for $t = 0$ **to** T_{max} **do**
  set $\epsilon_t \leftarrow \frac{2}{2+t}$;
  define $\delta_i := \boldsymbol{x}_i^\top \left(\boldsymbol{X}^\top \operatorname{diag}(z^t)\boldsymbol{X}\right)^{-1} \boldsymbol{x}_i$;
  find $z^\star$ solution of $\max_{z\in[0,1]^n : z^\top \boldsymbol{e} \le d} \boldsymbol{\delta}^\top z$ via sorting ;
  update $z^{t+1} \leftarrow (1-\epsilon_t)z^t + \epsilon_t z^\star$;
  if $(\boldsymbol{\delta}^\top z^\star - p) \le \epsilon_{target} |f(z^t)|$ **then**
    break;
  end
end
return z^t.

Interior point method We use the commercial solver Mosek to solve the SDP primal formulation (EC.2) or its equivalent dual formulation (EC.3). We set the feasibility tolerance to 10^{-6}.

Outer approximation We implement an outer-approximation algorithm, using Gurobi to solve the linear optimization problem at each iteration. We implement the stabilization/acceleration technique presented in Fischetti et al. (2017) at the root node.

Dual subgradient descent The algorithm solves (EC.4) and proceeds like a regular gradient descent, using subgradients instead of gradients. The maximization problem requires $O(np^2)$ operations to compute the n entries $(\boldsymbol{x}_i^\top \boldsymbol{A}\boldsymbol{x}_i)$, $i = 1, \ldots, n$, and $O(n\log(n))$ operations to sort them. The former constitutes the most time-consuming step of the algorithm (more than computing $\boldsymbol{A}^{-1}$ since $n \ge p$). The algorithm has two stopping criteria: a bound on the total number of iterations, $T_{\texttt{max}}$ (equivalently, we could impose a time limit) and a target optimality gap, $\epsilon_{\texttt{target}}$.

Regarding the step size at each iteration t, $\delta^{(t)}$, we experience the fastest convergence with a rule of the form $\boldsymbol{\delta}^{(t)} := \eta_t \left(\boldsymbol{A}^{(t)}\right)^2$, which resembles a Newton step (the Hessian of $q(\boldsymbol{A})$ is approximately

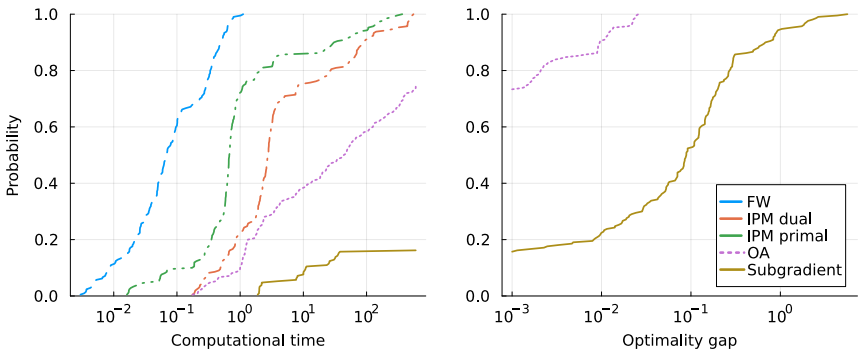


Figure EC.1 **Cumulative distribution of the computational time (left panel) and optimality gap at termination (right panel) achieved by five different algorithms for solving the Boolean relaxation** (EC.1)**. Results are obtained on the same instances as those presented in Section 4.3.**

$\boldsymbol{A}^{-2}$). We take $\eta_t = 10^{-3}$ and iteratively halve it to ensure that $\boldsymbol{A}^{(t+1)}$ remains semidefinite positive. We also consider a restarting scheme to accelerate convergence.

EC.2.3. Numerical evaluation

In this section, we compare numerically Frank-Wolfe algorithm on the primal formulation (FW in short), Mosek on the primal and dual SDP formulation respectively (IPM primal and IPM dual), an outer-approximation (a.k.a. Kelley's algorithm) on the primal formulation (OA), and a subgradient algorithm on the dual formulation (Subgradient). We impose a 10-minute time limit and 0.1% target optimality gap for all methods. We use the same instances as in Section 4.3.

Figure EC.1 represents the cumulative distribution of computational time (left panel) and optimality gap at termination (when available, right panel) for each of the five algorithms. Overall, it is clear that Frank-Wolfe largely outperforms all other approaches (by 1–2 orders of magnitude) and solve all instances in less than a second. As a result, we use this method in all our subsequent experiments.

Appendix EC.3: Algorithms for Generating Feasible Solutions

In this section, we provide further implementation details on existing methods to generate high-quality feasible solutions to Problem (1) and evaluate them numerically. We consider a linearization heuristic, sampling-based approaches, and local search. Based on our numerical experiments (see Section EC.3.4), we recommend solving the relaxation and applying greedy rounding to obtain a high-quality solution quickly (on average, 5.78% optimality gap in 0.149 seconds), and further refining this solution via local search at the root node.

EC.3.1. A linearization heuristic

We introduce a simple linearization heuristic to approximate (1) as a simple binary linear optimization problem. Although quite loose, this approximation can be useful to quickly initialize algorithms that require an initial feasible solution, e.g., local search algorithms, and can be generalized in cases where additional operational constraints are imposed on $\boldsymbol{z}$.

For any $p \times p$ matrix $\boldsymbol{W} \succeq \boldsymbol{0}$, the analytic-geometric mean inequality yields

$$\frac{1}{p}\operatorname{logdet}(\boldsymbol{W}) \leq \log(\operatorname{tr}(\boldsymbol{W})) - \log(p),$$

so that we can use the solution of the binary linear optimization problem

$$\max_{\boldsymbol{z}\in\{0,1\}^n} \quad \operatorname{tr}\left(\boldsymbol{X}^\top \operatorname{diag}(\boldsymbol{z})\boldsymbol{X}\right) \quad \text{s.t} \quad \boldsymbol{z}^\top \boldsymbol{e} \leq d,$$

to obtain a feasible solution to (1) and a valid upper-bound. Note that the linear optimization problem above can be solved efficiently by computing the variance $\boldsymbol{x}_i^\top \boldsymbol{x}_i$ at each location $i = 1, \ldots, n$, and taking the d largest values.

EC.3.2. Sampling-based algorithms

The first category of methods takes the fractional solution $\hat{\boldsymbol{z}} \in [0,1]^n$ obtained from solving the Boolean relaxation and generates a feasible binary solution. For example, one can greedily round the largest coordinates of $\hat{\boldsymbol{z}}$ to 1 until we saturate the budget constraint $\boldsymbol{e}^\top \boldsymbol{z} = d$ (as done, e.g., in Bertsimas et al. 2021). However, this approach is limited as it only generates one candidate solution and is highly dependent on the fractional solution $\hat{\boldsymbol{z}}$.

To overcome this limitation, sampling schemes have been investigated. A first sampling scheme (which we later refer to as Bernoulli sampling) consists of sampling each coordinate $\tilde{z}_i \sim \text{Bern}(\hat{z}_i)$ independently. In its original description (Raghavan and Tompson 1987), Bernoulli sampling simply rejects the sampled solution $\tilde{\boldsymbol{z}}$ whenever it is not feasible. Instead, we randomly turn on (or off) coordinates of $\tilde{\boldsymbol{z}}$ until the budget constraint is met exactly, $\boldsymbol{e}^\top \tilde{\boldsymbol{z}} = d$. The stopping criteria are a limit on total time, total number of iterations ($T_{\texttt{max}}$), and total number of iterations without improvement ($T_{\texttt{improv}}$).

Alternatively, for the problem of optimal experimental design specifically, Singh and Xie (2020) propose an alternative correlated sampling scheme. Their algorithm has time complexity $O(n^2)$ and achieves an $(1/e)$-approximation guarantee. We refer to Singh and Xie (algorithm 1, 2020) for implementation details[4]. We implement this sampling scheme with the same stopping criteria as Bernoulli sampling.

EC.3.3. Local search procedure

A second category of approaches is based on local search, often referred to as Fedorov's exchange method (see, e.g., Nguyen and Miller 1992) and studied in Madan et al. (2019). Given a feasible solution $\boldsymbol{z}$, the algorithm considers all pairs of indices (i, j) such that $z_i = 1$ and $z_j = 0$ and evaluates whether swapping them (i.e., setting $z_i = 0$ and $z_j = 1$ instead) would benefit the objective. To efficiently evaluate the change of objective for such a swap, it utilizes the Sherman–Morrison formula on determinant as follows: Denoting $\Lambda := (\boldsymbol{X}^\top \operatorname{diag}(\boldsymbol{z})\boldsymbol{X})^{-1}$, we have

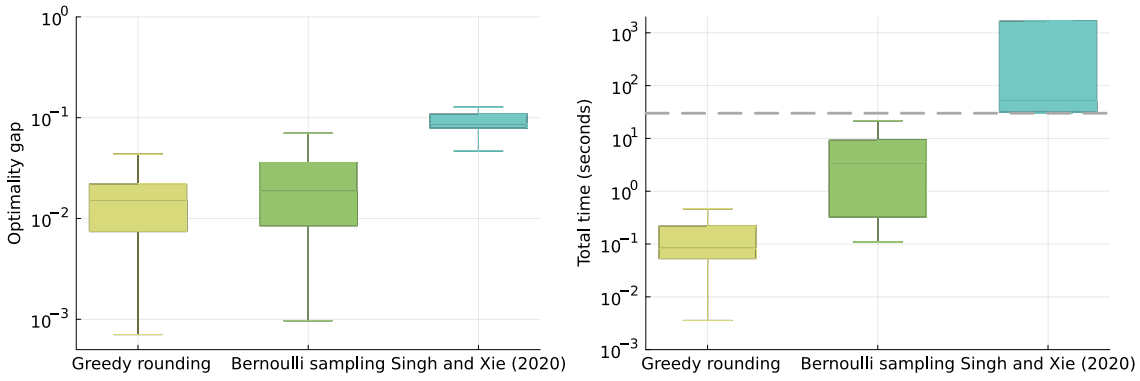
$$\begin{aligned}
&\det\left(\boldsymbol{X}^\top \operatorname{diag}(\boldsymbol{z})\boldsymbol{X} - \boldsymbol{x}_i\boldsymbol{x}_i^\top + \boldsymbol{x}_j\boldsymbol{x}_j^\top\right) > \det\left(\boldsymbol{X}^\top \operatorname{diag}(\boldsymbol{z})\boldsymbol{X}\right) \\
\Leftrightarrow\ &\det\left(\boldsymbol{I}_2 + [\boldsymbol{x}_i; \boldsymbol{x}_j]^\top \boldsymbol{\Lambda} [-\boldsymbol{x}_i; \boldsymbol{x}_j]\right) > 1 \\
\Leftrightarrow\ &(1 - \boldsymbol{x}_i^\top \Lambda \boldsymbol{x}_i)(1 + \boldsymbol{x}_j^\top \Lambda \boldsymbol{x}_j) + (\boldsymbol{x}_i^\top \boldsymbol{\Lambda} \boldsymbol{x}_j)^2 > 1.
\end{aligned}$$

Notice that, at each iteration, the local search algorithm needs to enumerate over $d \times (n - d)$ swap pair. The algorithm terminates after a maximum number of swaps or when no objective-improving swaps can be found for the current incumbent.

EC.3.4. Numerical evaluation

We compare the performance and scalability of different methods for obtaining high-quality solutions. To do so, we consider the same 21×10 instances generated in Section 4.3. Precisely, we generate various OED instances with $n \in \{100, 200, 500, 1000, 2000, 5000, 10000\}$, $p = 20$, and $d/p = 4$; $n = 1000$, $p \in \{5, 10, 15, 20, 40, 50, 60\}$, and $d/p = 4$; and $n = 1000$, $p = 20$, and $d/p \in \{1, 2, 3, 4, 5, 7, 10\}$. We generate 10 random instances for each parameter combination.

The first category of methods takes the relaxed solution as input and generates a feasible binary solution via rounding or sampling. We impose a time limit of 30 seconds for all methods. For sampling-based methods, we terminate after 10,000 iterations or 1,000 iterations without improvement in solution quality. Figure EC.2 compares the distribution of optimality gaps (left panel) and total computational time (i.e., including the time needed to solve the relaxation, right panel) for three methods: Greedy rounding, Bernoulli sampling, and the sampling of Singh and Xie (2020). In terms of solution quality, we observe that greedy rounding and Bernoulli sampling achieve comparable performance, with a small edge for greedy rounding (5.78% and 6.16% average optimality gap). However, greedy rounding requires one order of magnitude less time than Bernoulli sampling (0.15 seconds vs. 6.06 seconds on average). On the other hand, we find that the sampling of Singh and Xie (2020) is significantly more expensive (1,619 seconds on average). For this reason, except for



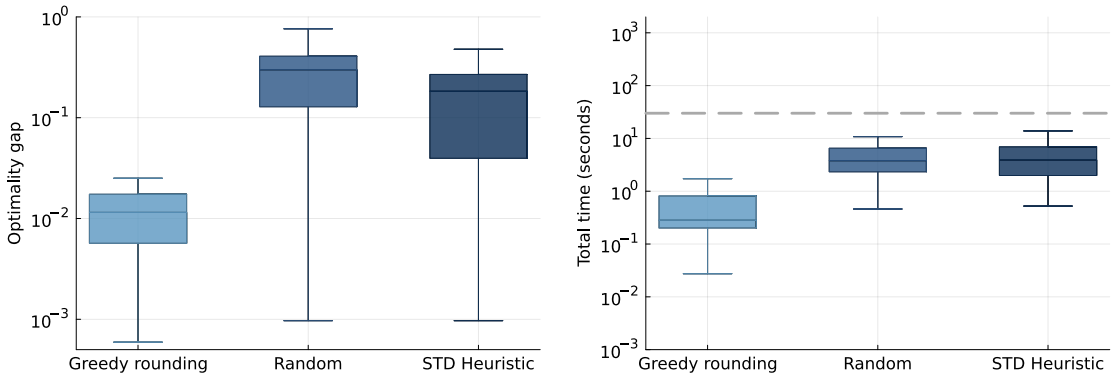
(a) Optimality gap (b) Total time (in seconds)

Figure EC.2 Box plot of optimality gap (left) and total time (right) obtained by each rounding/sampling method over 210 instances. The grey dashed line represents the 30-second time limit.

very small instances ($n \leq 500$, $p = 20$), the algorithm exceeds the time limit after the first iteration and does not achieve satisfactory performance (80.86% average optimality gap). Based on these observations, we conclude that greedy rounding is the most effective method to obtain a binary solution from the relaxation.

The second category of methods is based on local search. We evaluate three initialization strategies: the greedily rounded solution from the relaxation (Greedy rounding); a randomly sample feasible solution (Random); the solution obtained by a linearization heuristic (STD, see Section EC.3.1). We impose a time limit of 30 seconds and stop after 1,000 iterations (or whenever no local improvement can be found). Results (optimality gap and total time) are presented in Figure EC.3. We observe that the choice of initialization point plays a critical role in the performance of local search: Greedy rounding initialization (light blue, left box) achieves one order of magnitude improvement over other methods, in terms of both solution quality (on average 3.95% optimality gap vs. 29.09% for Random and 17.76% for STD Heuristic) and numerical tractability (on average 0.6 seconds vs. 5.67s for Random and 5.64s for STD Heuristic).

We should emphasize that we report total computational time, i.e., including the time required to solve the relaxation if needed. If we break-down this total computational time, we observe that greedy rounding requires more time to generate the initial solution (0.149 seconds on average vs. $\leq$0.001 seconds for the other two initializations). However, the local search algorithm converges 10 times faster with this initialization, thus largely making up for the increased time spent on constructing that initial solution.



(a) Optimality gap (b) Total time (in seconds)

Figure EC.3 Box plot of optimality gap (left) and total time (right) obtained by local search with three different initialization strategies over 210 instances.

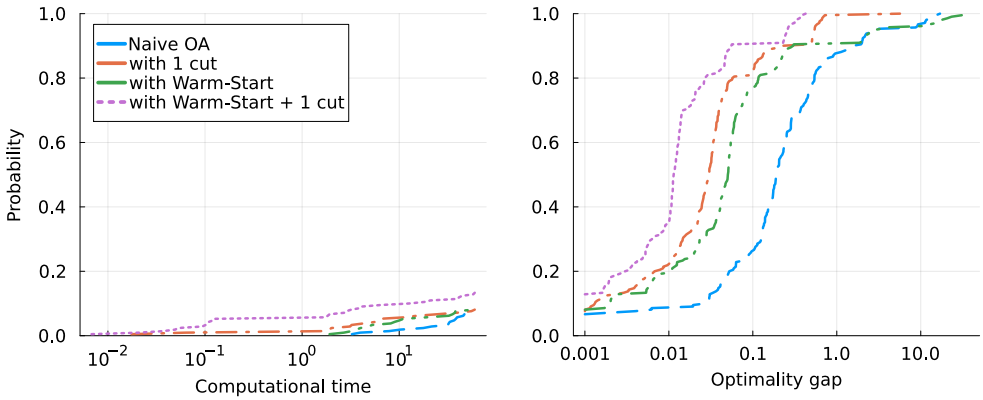
Overall, we recommend solving the relaxation and applying greedy rounding to obtain a high-quality solution quickly (on average, 5.78% optimality gap in 0.149 seconds). Time permitting, this solution can be further improved by serving as the initialization of a local search scheme. However, we should highlight that the improvement in solution quality (3.95% vs 5.78% on average) can be considered moderate with regard to the increase in average computational cost (+0.481 seconds, i.e., $\times 3$).

Appendix EC.4: Additional Numerical Results on Synthetic OED Data

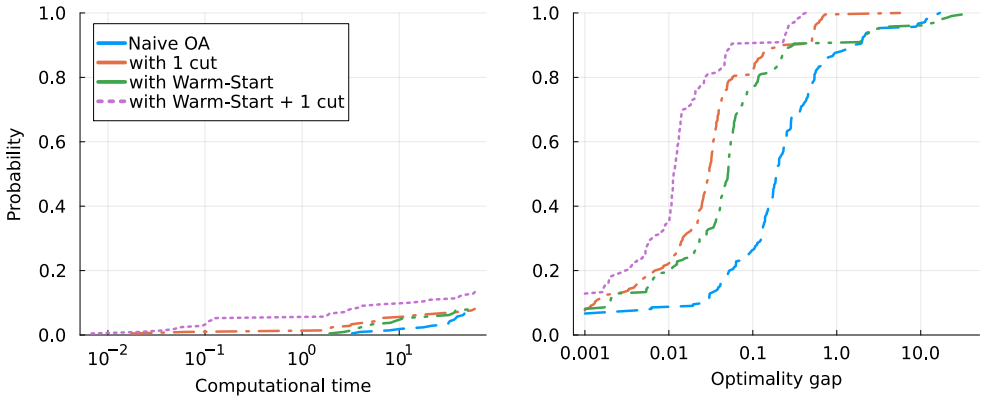
EC.4.1. Implementation of the benchmark (outer approximation)

We compare different implementations of the outer-approximation scheme, on the same instances as those used in 4.3. Figure EC.4 reports the performance (summarized by the cumulative distribution of computational time and optimality gap at termination) of four different implementations: the naive outer-approximation scheme implemented via lazy callbacks ('Naive'); a version where the linear approximation obtained from the relaxation is provided at the root node ('with 1 cut'); a version where the solution from rounding the relaxation followed by local search is provided as a warm-start ('with Warm-Start'); and a version where we do both of the aforementioned strategies ('with Warm-Start + 1 cut'). We consider both CPLEX 22.1 (top panel) and Gurobi 10.0 (bottom panel) as commercial solvers We find that the three enhanced strategies improve upon the naive implementation.

Overall, we observe that the combination of adding cut and warmstarting the lower bound dominates all the other strategies, hence is the one we consider as a benchmark in our numerical experiments. With these enhancements, the performance of CPLEX marginally better.



(a) Using CPLEX



(b) Using Gurobi

Figure EC.4 **Cumulative distribution of computational time and final gap after reaching the time limit (60 seconds) for different implementations of the OA algorithm. Left panel: percentage of instances solved after a certain amount of time. Right panel: distribution of optimality gap upon termination.**

EC.4.2. Screening rule

Figure 1 demonstrates that the effectiveness of our screening rule depends on the accuracy at which the relaxation is solved. This benefit could be due to the fact that solving the relaxation more accurately leads to a tighter upper bound (UB) or a higher-quality binary solution (LB). To appreciate the contribution of each mechanism, we consider applying the screening rule with the linear upper approximation obtained from the relaxation solved at 10^{-4}-optimality with the binary solution found from the 10^{-2}-optimal solution (via rounding followed by local search, and vice versa). The performance of each approach is reported in Figure EC.5. Starting from the solution $(10^{-2},10^{-2})$, we observe that improving the quality of the UB has a greater effect on the power of the screening rule, with almost no difference depending on the relaxation used to find a high-quality

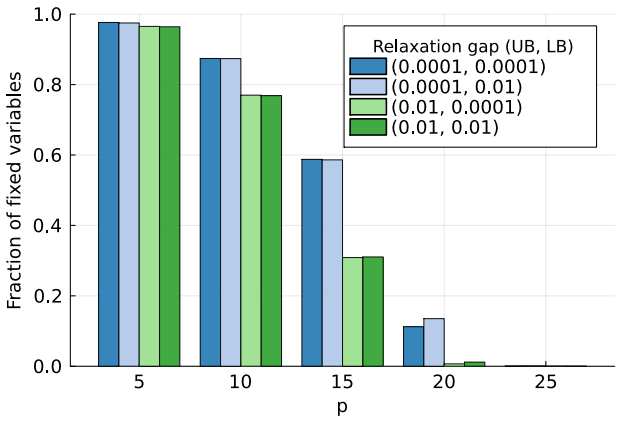


Figure EC.5 **Fraction of entries fixed by the screening rule, depending on the optimality gap of the solution from relaxation used to generate the linear upper-approximation (UB) and that of the solution used to generate a good feasible solution (LB), averaged over 5 OED instances with $n = 1,000$, $d = 4 \times p$ and increasing p.**

solution. This observation suggests that the feasible solution heuristic (greedy rounding followed by local search) is quite robust to inaccuracies in the relaxed solution.

EC.4.3. Implementation of our tailored branch-and-bound

The pseudo-code of our tailored branch-and-bound algorithm is given in Algorithm EC.4.1. Our algorithm involves several design decisions, in particular regarding the effort required at each node and the branching rule.

For the effort required at each node, the minimum we have to do is to solve the Boolean relaxation for the subproblem. Given the relaxed solution, finding a feasible solution via greedy rounding/sorting is not computationally expensive so we do it each time. However, we know that this feasible solution can be improved by local search. We consider applying this local search procedure at the root node only or at each node. Furthermore, with the information from the relaxation (i.e., a tight linear upper-approximation of the objective) and a good solution, we can apply our screening rule. Again, we consider applying this screening rule at the root node only or at every node.

Regarding the branching rule, we consider two options: Branching on the most fractional entry, i.e., $i^\star \in \arg\max_i \bar{z}_i(1-\bar{z}_i)$, or branching on the entry with the highest marginal objective increase, i.e., $i^\star \in \arg\max_i \{|\nabla f(\bar{z}_i) : \bar{z}_i(1-\bar{z}_i) > 0\}$.

Based on separate experiments conducted on the same instances as those used in Section 4.3, we find that applying local search or the screening rule at each node does not bring additional benefits compared with applying them at the root node only .Regarding the branching decision, branching

Algorithm EC.4.1: Tailored branch-and-bound scheme

input Target gap ϵ;
⋮
Initialize $\mathcal{A} \leftarrow \{\mathcal{P}(\emptyset, \emptyset)\}$;
Initialize $(UB, LB) \leftarrow (+\infty, -\infty)$, $z^\star \leftarrow \mathbf{0}$;
while $\mathcal{A} \neq \emptyset$ *and* $UB - LB > \epsilon|LB|$ **do**
  Pop `node` (and associated $\mathcal{P}(\mathcal{I}_0, \mathcal{I}_1)$) from $\mathcal{A}$ with highest upper bound (`node selection rule`) ;
  if $|\mathcal{I}_1| > d$ *(*`node` *infeasible)* **then**
    Set `node.ub` $\leftarrow -\infty$;
  else if $\{1, \ldots, n\} \setminus \mathcal{I}_0 \leq d$ **then**
    Set $\bar{z}_i = 1$ if $i \in \{1, \ldots, n\} \setminus \mathcal{I}_0$, 0 otherwise ;
    Set `node.ub` $= f(\bar{z})$, `node.lb` $= f(\bar{z})$;
    if *node.lb* $> LB$ **then**
      Set $LB \leftarrow f(\bar{z})$, $z^\star \leftarrow \bar{z}$;
    end
  else
    Solve the Boolean relaxation of $\mathcal{P}(\mathcal{I}_0, \mathcal{I}_1)$ using FW (Algorithm EC.2.1), obtain $\bar{z}$;
    Set `node.ub` $\leftarrow f(\bar{z})$;
    if *node.ub* $> LB$ **then**
      Obtain z_0 feasible for $\mathcal{P}(\boldsymbol{\ell}, \boldsymbol{u})$ by greedily rounding $\bar{z}$;
      At the root node: Improve z_0 by conducting local search ;
      Set `node.lb` $\leftarrow f(z_0)$;
      if *node.lb* $> LB$ **then**
        Set $LB \leftarrow f(z_0)$, $z^\star \leftarrow z_0$;
      end
      At the root node: Apply the screening rule (Proposition 1), update $(\mathcal{I}_0, \mathcal{I}_1)$;
      Find $i^\star \in \arg\max_i \{|\nabla f(\bar{z}_i)| : \bar{z}_i(1 - \bar{z}_i) > 0\}$ (`branching rule`) ;
      Define `left` and `right` child nodes, associated with $\mathcal{P}(\mathcal{I}_0 \cup \{i^\star\}, \mathcal{I}_1)$ and $\mathcal{P}(\mathcal{I}_0, \mathcal{I}_1 \cup \{i^\star\})$ respectively ;
      Set `left.ub` = `right.ub` = `node.ub` ;
      Set `left.lb` = `node.lb` if $z_{0,i^\star} = 0$ or `right.lb` = `node.lb` if $z_{0,i^\star} = 1$;
      Push `left` and `right` to $\mathcal{A}$;
    end
  Filter $\mathcal{A} \leftarrow \{\mathtt{a} \in \mathcal{A} : \mathtt{a.ub} > LB\}$;
  Update $UB = \max\{\mathtt{a.ub} : \mathtt{a} \in \mathcal{A}\}$;
end

on the fractional entry where the objective function has the highest derivative appears noticeably more efficient than picking the most fractional entry.

Note that the general idea of embedding Frank-Wolfe within a branch-and-bound as the node relaxation solver for mixed-integer convex optimization is the underpinning idea of the open-

source solver `Boscia.jl` (Hendrych et al. 2025). While faster than other open-source approaches (Hendrych et al. 2024), we have found it slower than outer approximation implemented with commercial solvers (see Figure EC.7.

EC.4.4. Scalability of discrete optimization algorithms

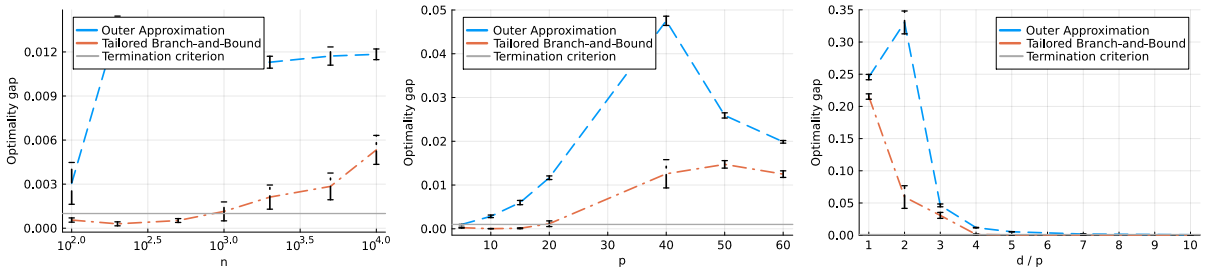
Figure EC.6 represents the average optimality gap at termination for the outer approximation and the tailored branch-and-bound algorithm as a function of n (fixing $p = 20$ and $d/p = 4$), p (fixing $n = 1000$ and $d/p = 4$), and d (fixing $n = 1000$ and $p = 20$). Overall, the behavior of the optimality gap is aligned with that of computational time in Figure 3: We observe a smooth increase for both algorithms as n increases; a non-monotonous inverted-U-shape behavior as a function of p; and a sharp transition from hard to easy instances as d/p increases.

We also compare our branch-and-bound code with `Boscia.jl`, an open-source mixed-integer optimization solver using Frank-Wolfe for each node relaxation. To the best of our knowledge, it is the best open-source solver for solving OED problems (see Hendrych et al. 2024). As reported in Figure EC.7, however, our tailored branch-and-bound code is significantly faster—probably because of the customized branching strategy and the novel screening rule.

Appendix EC.5: Additional Numerical Results on Real-World Case Study

EC.5.1. Additional numerical results to Section 6.2

Figure 5 represents the cumulative distribution of time (left panel) and optimality gap at termination (right panel) for OA methods and our branch-and-bound algorithm. In our setting, the main dimension impacting computational difficulty is the length of the route d. With routing constraint, n naturally scales with d (as d^2) and we observe that the difficulty of the problem (measured in terms of computational time, gap at termination, and fraction of instances solved at optimality) increases with n (or d), as evidenced in Figure EC.8.



(a) Varying n (b) Varying p (c) Varying d/p

Figure EC.6 **Average computational time of both discrete optimization approaches when varying n (left), p (middle), and d/p ratio (right). The results are averaged over 10 simulations.**

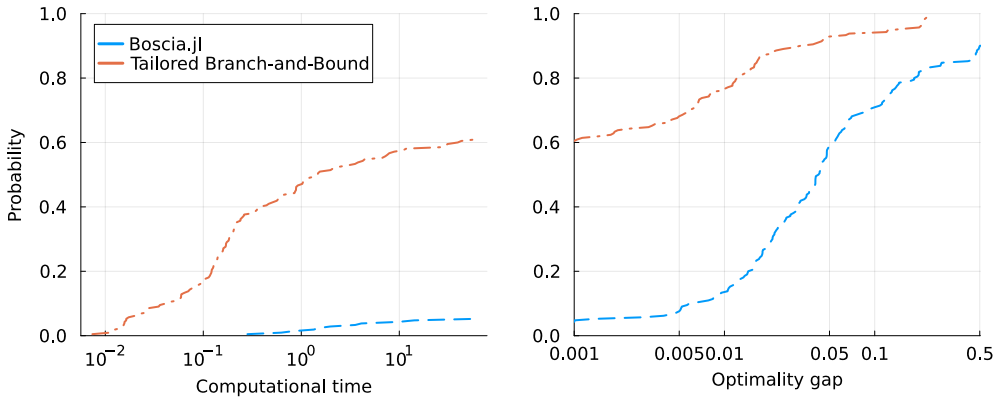
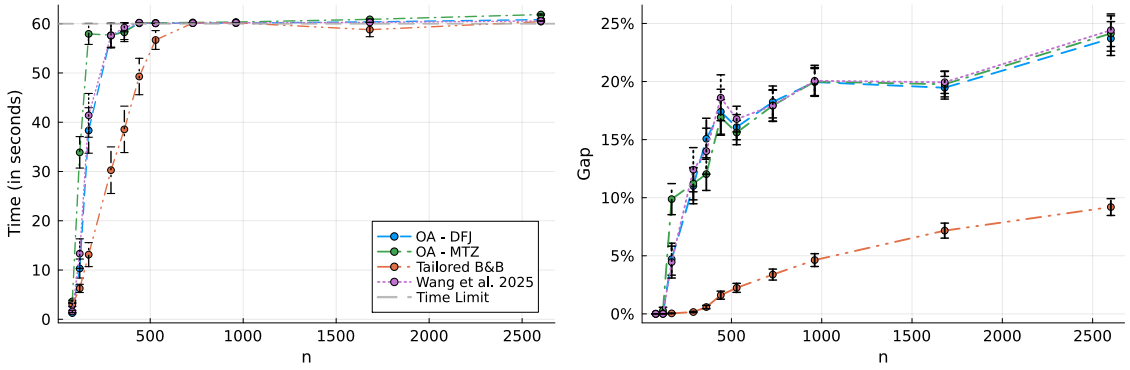
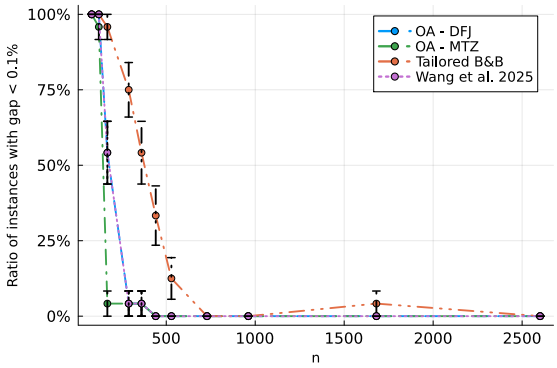


Figure EC.7 **Cumulative distribution of computational time (in seconds) and final gap after reaching the time limit (60 seconds) for Boscia.jl and our tailored branch-and-bound algorithm on 210 OED instances.**



(a) Computational time (in seconds)

(b) Optimality gap (in %)

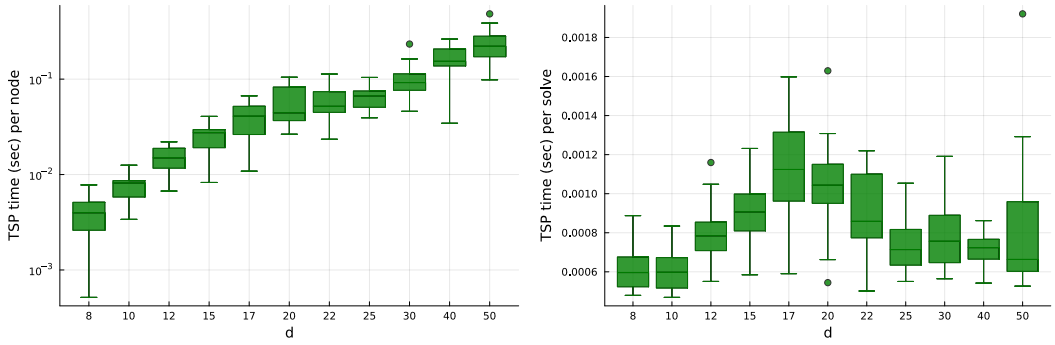


(c) Fraction of instances solved to optimality (in %)

Figure EC.8 **Convergence metrics for OA schemes and our tailored branch-and-bound (B&B), on OED instances with routing constraints, as the number of locations n increases ($n = O(d^2)$).**

Figure 8 illustrates the computational efficiency of our screening rule by reporting the number of constraints generated per subproblem/node (left panel) and per TSP instance (right panel). Figure EC.9 complements this picture by reporting computational time.

The inspection we do as part of the screening rule serves as the first step to finding high-quality solutions. For this reason, the quality of our heuristic depends on the depth of the node, and in particular, the number of fixed locations $|\mathcal{I}_0 \cup \mathcal{I}_1|$. For each instance that is solved to optimality, we record the fraction of fixed locations $|\mathcal{I}_0 \cup \mathcal{I}_1|/n$ (left panel) and the fraction of visited locations $|\mathcal{I}_1|/d$ (right panel) at the node where we find the optimal solution, and represent their distribution across instances in Figure EC.10. We observe that our heuristic mostly finds the best solution when more than 75% of the locations are fixed (i.e., either visited or excluded) and when more than 50% of the trajectory is determined ($|\mathcal{I}_1|/d > 0.5$).



(a) Per subproblem/node (b) Per TSP instance

Figure EC.9 **Distribution (boxplot) of the TSP computational time in each subproblem/node of the branch-and-bound tree (left panel) and per TSP instance solved (right panel) as the problem dimension d increases.**

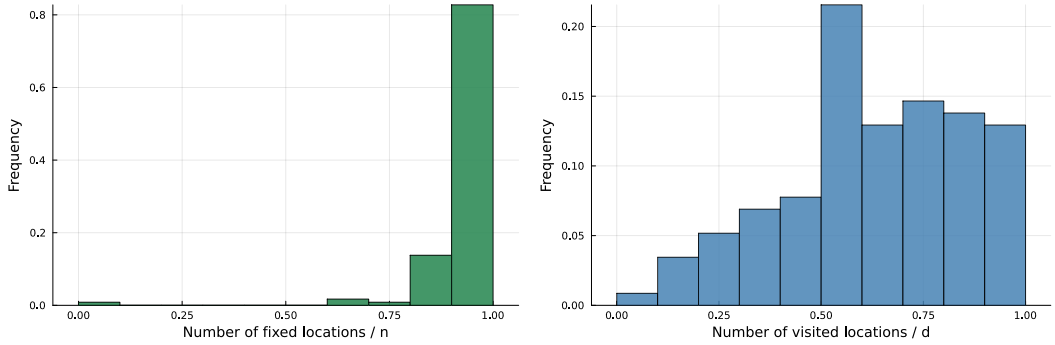
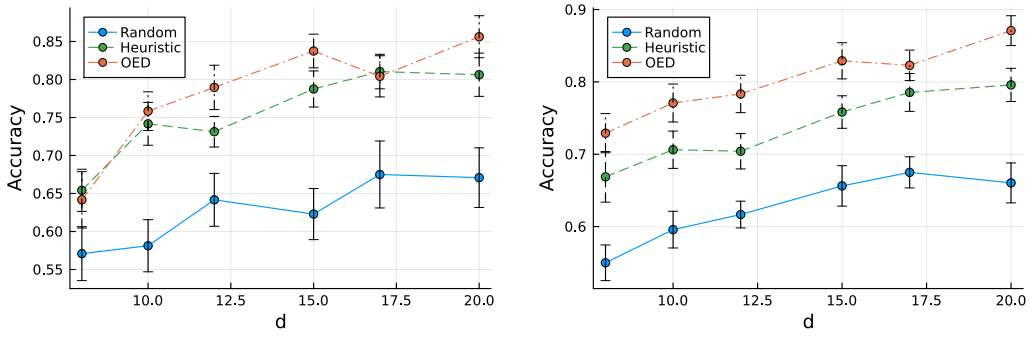


Figure EC.10 **Distribution of the fraction of fixed locations, $|\mathcal{I}_0 \cup \mathcal{I}_1|/n$, (left panel) and fraction of visited locations, $|\mathcal{I}_1|/d$, (right panel) when finding the best feasible solution. Results are aggregated over all instances that are solved to optimality (116 out of 264 instances).**

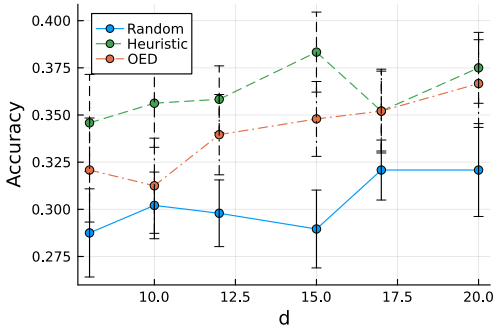
EC.5.2. Additional numerical results to Section 6.3

In Section 6.3, we evaluate the impact of using our OED formulation on downstream prediction accuracy by reporting the prediction error $(1-R^2)$ of models obtained using different data collection strategy. Alternatively, we could measure the accuracy in terms of ability of picking the right model, i.e., identify $\hat{j}$ minimzing the average prediction error $(y-x_j)^2$ over the sampled location and compare whether $\hat{j}$ matches $j^\star$. Figure EC.11 reports the same results as Figure 9-10 in terms of accuracy instead of errors. Similarly, we observe that the OED solution provides significant improvements over the STD heuristic and the random design solution, under correct specification as well as non-i.i.d. noise. In the most extreme case of misspecification (non-linearity and non-i.i.d. noise), we observe that the STD heuristic leads to the most accurate model picking. However, in this case, the model $j^\star$ (which is used to generate y) might not be the most accurate, so the notion of a 'true' model is challenged.



(a) Specified

(b) Misspecified: Non i.i.d. noise



(c) Misspecified: Non-linear + non i.i.d. noise

Figure EC.11 **Average model picking accuracy for the downstream regressor depending on the method used for collecting data: random, a linearization heuristic (STD, see Section EC.3.1), or our tailored branch-and-bound. For each value of d, results are averaged over 24 instances and 20 generations of ground true data.**

References

Bertsekas D (2016) *Nonlinear Programming*, volume 4 (Athena Scientific).

Bertsimas D, Cory-Wright R, Pauphilet J (2021) A unified approach to mixed-integer optimization problems with logical constraints. *SIAM Journal on Optimization* 31(3):2340–2367.

Bertsimas D, Pauphilet J, Van Parys B (2020) Sparse regression: Scalable algorithms and empirical performance. *Statistical Science* 35(4):555–578.

Fischetti M, Ljubić I, Sinnl M (2017) Redesigning Benders decomposition for large-scale facility location. *Management Science* 63(7):2146–2162.

Hendrych D, Besançon M, Pokutta S (2024) Solving the optimal experiment design problem with mixed-integer convex methods. *22nd International Symposium on Experimental Algorithms*.

Hendrych D, Troppens H, Besançon M, Pokutta S (2025) Convex mixed-integer optimization with frank–wolfe methods. *Mathematical Programming Computation* 1–27.

Joshi S, Boyd S (2008) Sensor selection via convex optimization. *IEEE Transactions on Signal Processing* 57(2):451–462.

Kelley JEJ (1960) The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics* 8(4):703–712.

Li Y, Fampa M, Lee J, Qiu F, Xie W, Yao R (2024) D-optimal data fusion: Exact and approximation algorithms. *INFORMS Journal on Computing* 36(1):97–120.

Li Y, Xie W (2024) Best principal submatrix selection for the maximum entropy sampling problem: scalable algorithms and performance guarantees. *Operations Research* 72(2):493–513.

Madan V, Singh M, Tantipongpipat U, Xie W (2019) Combinatorial algorithms for optimal design. *Conference on Learning Theory*, 2210–2258 (PMLR).

Nguyen NK, Miller AJ (1992) A review of some exchange algorithms for constructing discrete D-optimal designs. *Computational Statistics & Data Analysis* 14(4):489–498.

Raghavan P, Tompson CD (1987) Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica* 7(4):365–374.

Singh M, Xie W (2020) Approximation algorithms for D-optimal design. *Mathematics of Operations Research* 45(4):1512–1534.