European Journal of Operational Research xxx (xxxx) xxx

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/eor

Stochastics and Statistics

Leveraging 0–1 linear programming for partitioning and sampling in simulation-based optimization: The case of nested partitions

Tommaso Schettini [a,*], Fausto Errico [b], Jorge E. Mendoza [c], Carolina Osorio [d]

[a] Concordia University, Department of Mechanical, Industrial and Aerospace Engineering, Montréal, H3G 1M8, Quebec, Canada
[b] École de technologie supérieure, Department of system engineering, Montréal, H3C 1K3, Quebec, Canada
[c] HEC Montréal, Department of Logistics and Operations Management, Montréal, H3T 2A7, Quebec, Canada
[d] HEC Montréal, Department of Decision Sciences, Montréal, H3T 2A7, Quebec, Canada

ARTICLE INFO

Dataset link: https://github.com/tms-publications/DSO-Nested-Clusters, https://arxiv.org/abs/2609.30628

Keywords:
Combinatorial optimization
Clustering
Nested partitions
Discrete simulation-based optimization

ABSTRACT

Discrete Simulation-based Optimization (DSO) problems typically involve highly complex stochastic systems such as urban road networks or electric grid networks. For such problems, computing the value of the objective function requires computationally intensive simulations. Many solution methods rely on iteratively sampling and partitioning the feasible space to progressively concentrate the sampling effort on its most promising portions. To boost the effectiveness of such methods, it is fundamental to quickly identify which portions of the feasible space are relevant to explore through an effective partitioning scheme. In this paper, we focus on binary DSO problems and introduce Nested Clusters, an enhancement of Nested Partitions that leverages the mathematical formulation of the feasible space to construct auxiliary 0–1 linear programming formulations for partitioning and sampling. Additionally, we expand Nested Clusters by introducing a biasing technique that improves the average quality of the sampled solutions, boosting the overall performance. Finally, we apply Nested Clusters to a simulation-based charger location problem for an autonomous fleet of ride-hailing taxis in New York City, which serves as our primary benchmark. We conduct further controlled synthetic tests on four binary simulation–optimization problems spanning different combinatorial structures and stochastic evaluators. On the tested ride-hailing charger-location instances, both proposed partitioning and biased sampling strategies independently improve solution quality relative to Nested Partitions; combining them yields an average 2.2% improvement in the best solutions found.

1. Introduction

With the ever-increasing availability of data and computational resources, intricate data-driven simulation-based models have become common tools for decision-makers to inform and optimize their decisions (Agalianos et al., 2020; Winkelhaus & Grosse, 2020). Embedding these simulation-based models within combinatorial optimization frameworks enables decision-makers to capture the dynamics of stochastic problems, leading to more realistic models. In many such problems, the decisions are discrete in nature. We refer to these as Discrete Simulation-based Optimization (DSO) problems. A representative setting is the design of charging and parking infrastructure for electric ride-hailing fleets, where binary location decisions must be evaluated through stochastic simulations of vehicle movements, customer demand, charging queues, and fleet operations.

Formally, DSO problems are discrete combinatorial optimization problems that use the output of a stochastic and often black-box simulator as part of the problem definition. This could be because the value of their objective function depends on the output of a simulator, or some of their constraints require a simulator to verify whether they are violated (Nelson, 2010). In addressing these problems, the majority of the solution algorithms in the literature, such as those introduced in Hong and Nelson (2006), Shi and Ólafsson (2000a), Xu and Nelson (2013) and Xu et al. (2013), employ a sampling-based approach, commonly known as Adaptive Random Search (ARS).

We note that within the DSO literature, the term “sampling” often accounts for both the act of generating solutions and the subsequent simulation step to evaluate them. For the sake of clarity, in this paper, we use the term sampling exclusively to refer to the generation of solutions and the term “evaluating” to refer to the act of simulating a solution to either estimate its objective function value or verify simulation-based constraints. ARS initially evaluates solutions uniformly sampled from the feasible space. The strategy then samples

* Corresponding author.
E-mail addresses: tommaso.schettini@concordia.ca (T. Schettini), fausto.errico@ets.ca (F. Errico), jorge.mendoza@hec.ca (J.E. Mendoza), carolina.osorio@hec.ca (C. Osorio).

https://doi.org/10.1016/j.ejor.2026.09.048
Received 21 August 2025; Accepted 29 September 2026
Available online 10 October 2026
0377-2217/© 2026 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

and evaluates additional solutions, using information from previously evaluated solutions to guide the sampling procedure (Nelson, 2010). Algorithms based on this strategy concentrate the sampling of solutions on a gradually narrowing promising region, i.e., a region that is expected to contain high-quality solutions. In particular, these procedures partition the feasible space into sub-regions and iteratively select the promising region from the set of candidates generated by partitioning.

Algorithms based on ARS mainly differ with respect to how they partition the feasible space and how they sample solutions. The design of these two components, partitioning and sampling, plays a vital role in determining the effectiveness of such methods. This is primarily due to the high computational time required by simulators, which limits the number of solution evaluations that may be conducted during the optimization process. It is therefore important to design a partitioning strategy ensuring that the identified sub-regions contain solutions that behave similarly with respect to the simulator (e.g., grouping solutions whose simulated objective functions are all within a small deviation from each other). Doing so allows the search to more confidently estimate the quality of the solutions in each sub-region, quickly identify high-quality sub-regions, and reduce computational time spent evaluating low-quality solutions. Similarly, it is important to design a sampling mechanism that is biased towards high-quality solutions, i.e., that assigns a high sampling probability to solutions that are likely to achieve a high objective function value. However, the sampling of solutions is complicated by the discrete constrained nature of the problem domain. Random combinations of variables are likely to result in infeasible solutions, and it is not possible to smoothly move from one feasible solution to another. This can potentially cause the sampling of solutions to become a highly time-consuming process. Finally, the design of these two components is intertwined. Since ARS algorithms primarily sample solutions within the promising region, the sampling strategy needs to be tailored to the structure of the promising region, as determined by the chosen partitioning strategy.

We note that, when all constraints are deterministically known, the feasible space of many DSO problems can be described through a mathematical program. While these formulations cannot be directly input into a solver to compute an optimal solution as they lack an analytical objective function, they may reveal useful information about the problem structure. To the best of our knowledge, no solution algorithm for DSO problems makes explicit use of mathematical programming formulations in their partitioning and sampling strategies.

In this paper, we specifically deal with combinatorial DSO problems where the feasible space is deterministically constrained and can be described using a 0–1 linear programming formulation. For this class of problems, we propose a mechanism that uses the mathematical structure of the feasible space of a DSO problem to construct auxiliary 0–1 linear programming formulations to inform the partitioning and sampling of a DSO method. Within this setting, we focus on Nested Partitions (NP) (Shi & Ólafsson, 2000a), an adaptive random search algorithm that iteratively partitions the feasible space and samples solutions from the resulting promising regions. We develop Nested Clusters (NC), an enhancement of NP that uses the 0–1 linear programming structure of the feasible space to guide partitioning and sampling under a limited simulation budget. 0–1 linear programming encompasses a large family of problems, including resource allocation, scheduling, and logistics. The general formulation of one such problem is as follows:

$$\min\ f(x) = \mathbb{E}_{\omega\in\Omega}[F(x,\omega)] \tag{1}$$

$$Ax \geq b \tag{2}$$

$$x \in \{0,1\}^n, \tag{3}$$

where x denotes the vector of n decision variables. Constraints (2) are linear constraints, where matrix A and vector b are deterministically known. We denote by $f(x)$ the performance function that evaluates the quality of a solution x, which corresponds to the expected value of a function $F(x,\omega)$ over the set of possible scenarios $\omega \in \Omega$. Note that we assume that $f(x)$ cannot be directly computed and is estimated through stochastic simulation. We denote by $\hat{f}(x)$ an estimate of $f(x)$. For consistency, we present the methodology using a minimization convention. Maximization problems are handled by applying the algorithm to the negated objective function; reported application-level objective values are subsequently transformed back to their original maximization convention.

NC introduces two strategies for this class of problems: (1) a cluster-based partitioning strategy, and (2) a 0–1 linear programming-based sampling strategy. For (1), we consider that the feasible space of a constrained optimization problem generally exhibits some form of structure, where solutions are unevenly distributed in the space of the variables. To detect this structure, we use k-means clustering to induce a partition of the feasible space, grouping together solutions that are "close" to each other with respect to the variables that describe them, e.g., have a small Euclidean distance between each other. To enhance this strategy, we devise a streamlined approach for incorporating problem-specific information into the clustering, by projecting the solutions onto an auxiliary space that is engineered to highlight the most relevant aspects of the solutions with respect to the problem at hand, and using these projected solutions to partition the space. For (2), we generate and solve an auxiliary formulation of the problem that encodes the feasible space of the problem but uses a linear randomized objective function. We refine this strategy with an additive bias term in the objective function of the sampling formulation aimed at increasing the likelihood of sampling high-quality solutions.

While these strategies are presented in the context of NP, they address two operations that are common to ARS algorithms: partitioning the search space and sampling solutions from selected regions. For this reason, their integration into other ARS frameworks represents a natural direction for future research.

To assess NC on a simulation-based charger location problem, we compare it to a standard implementation of NP. We consider the problem of locating charging and parking infrastructure for a fleet of electric vehicles operating a ride-hailing service. Specifically, we test the proposed strategies on a New York City case study with realistic demand estimated from real field data (DIMACS, 2022). Results show that, on the tested case study, NC outperforms NP, and each proposed component yields an improvement in the average objective values of the sampled solutions, and the objective value of the best sampled solutions, compared to the baseline implementation of NP. The best results are obtained when both the cluster-based partitioning and the model-based sampling are applied in conjunction, leading to an average improvement of 2.2% in the profit achieved by the fleet as a result of the improved charger location. We conduct further controlled synthetic tests on four binary simulation–optimization problems spanning different combinatorial structures and stochastic evaluators, which are presented in Section 2 of the supplementary material. The remainder of this paper is structured as follows. In Section 2 we present a review of relevant literature. In Section 3 we discuss the proposed framework in detail. In Section 4 we present the case study of locating charging and parking infrastructure for a ride-hailing fleet in New York City. In Section 5 we present our computational results. The additional synthetic experiments are presented in Section 2 of the supplementary material. Lastly, in Section 6 we present concluding remarks.

2. Literature review

Simulation-based optimization comprises a broad class of optimization problems (Nelson, 2010). We focus our literature review on simulation-based optimization problems with discrete variables, i.e., DSO problems. For a more general review of simulation-based optimization, we refer the reader to de Sousa Junior et al. (2019).

When addressing DSO problems, several authors have opted to use tailored metaheuristics to explore the solution space. The simulator is then used as a black-box to evaluate the objective function and/or the

simulation-based constraints of the solutions found by the search (Can et al., 2008; Costa et al., 2015; Gan et al., 2019; Heinzl & Kastner, 2020; Javanroodi et al., 2019; Uhlig & Rose, 2011). Such approaches tend to be effective in cases where the used simulator is not very computationally expensive, which affords them time to evaluate a sufficiently large number of solutions during the search. To this end, genetic algorithms represent one of the most commonly applied metaheuristic approaches (Can et al., 2008; Gan et al., 2019; Javanroodi et al., 2019). Other metaheuristics have also been successfully employed. These include simulated annealing (Uhlig & Rose, 2011), tabu search (Costa et al., 2015), and variable neighborhood search (Heinzl & Kastner, 2020).

Several authors have developed general-purpose methods for DSO problems. Contrary to tailored metaheuristics, these methods make few, if any, assumptions about the problem structure. One of the main families of general-purpose methods is *ranking and selection* (R&S). These methods are aimed at identifying the best solution over a finite set of candidates by judiciously allocating simulation budget to the various alternatives (Chick & Inoue, 2001; Nelson et al., 2001). Within this family, optimal computing budget allocation (OCBA) methods allocate replications asymptotically to maximize the probability of correct selection under a fixed simulation budget (Chen & Lee, 2011; Chen et al., 2000). Because R&S and OCBA methods rely on an explicit candidate set, they are mainly applicable to problems with a limited solution space or to settings where a finite set of alternatives has already been generated. Thanks to recent advances in parallel computing they have been successful on specific large-scale problems with up to 10,000 feasible solutions (Pei et al., 2018). However, these methods meet their limit when the algorithm must search directly over a large combinatorial feasible region.

For such cases, researchers have turned to general-purpose random search methods that do not require complete enumeration of the candidate set. One class of such methods combines local random search with mechanisms that reduce the cost of noisy comparisons. Stochastic annealing algorithms explore the feasible space through local moves, while using probabilistic acceptance rules to avoid premature convergence to poor local optima (Alrefaei & Andradóttir, 1999; Ball et al., 2018; Branke et al., 2008). In this context, sample reuse provides a way to exploit simulation information collected at previous iterations, thereby reducing the number of additional simulation calls required to compare neighboring solutions. However, reusing samples may introduce conditional dependencies and bias in solution-quality estimates, which can distort the acceptance logic of the search (Eckman & Feng, 2018; Eckman & Henderson, 2018; Kiatsupaibul et al., 2018). Ball et al. (2024) address this issue for stochastic annealing by developing product-space algorithms that allow samples to be reused while preserving detailed balance.

Another class of general-purpose methods relies on adaptive random search (ARS). NP (Shi & Ólafsson, 2000a) uses this strategy. The main idea behind NP is to explore the search space by recursively partitioning it. NP iteratively designates a promising sub-region of the search space, partitions it, and evaluates randomly sampled solutions from each sub-region of the promising region. Finally, the algorithm updates the promising region based on the evaluated solutions.

Another general-purpose algorithm building on the ARS principle is Empirical Stochastic Branch and Bound (ESBB) introduced by Xu and Nelson (2013). ESBB uses a recursive partitioning strategy similar to that of NP, but encodes the nested partitions using a branch-and-bound tree. Rosen et al. (2016) extended ESBB to operate in a parallel computing environment, increasing the size of problems it can tackle. Both NP and ESBB achieve asymptotic convergence to the global optimum. Some authors have focused on algorithms that are guaranteed to converge to a local optimum in finite time. The COMPASS (Convergent Optimization via Most-Promising-Area Stochastic Search) algorithm falls into this category (Hong & Nelson, 2006).

Similar to NP, COMPASS uses the concept of promising region to focus the sampling effort. Specifically, COMPASS uses a Voronoi diagram to partition the feasible space using the set of evaluated solutions as generators. Therefore, the most promising region corresponds to the region of points whose Euclidean distance to the best evaluated point is smaller compared to all other evaluated points. A similar algorithm, the Adaptive Hyperbox Algorithm (AHA), was presented by Xu et al. (2013). In AHA, the most promising region is constructed as the largest hyperbox in the variable space that contains the best sampled point and excludes all other sampled points. Another algorithm with a guarantee of convergence to a local optimum is the R-SPLINE algorithm (Wang et al., 2013). The algorithm uses past evaluations of the simulated objective to construct a linear estimator of the simulated objective. Then, R-SPLINE uses this estimator to compute phantom gradient information and perform the search using a gradient descent strategy.

Several DSO algorithms rely on evaluating randomly sampled solutions to explore the feasible space of the problem (Hong & Nelson, 2006; Shi & Ólafsson, 2000a; Xu et al., 2013). Each of those algorithms identifies a promising region from which solutions are sampled. Thus, sampling solutions requires sampling from a constrained combinatorial optimization problem incorporating the constraints that identify the promising region, as well as any constraint associated with the feasible space of the problem. In the case of NP, no general-purpose sampling strategy is prescribed. Thus, sampling is defined on a case-by-case basis. As an example, in the implementation of NP for the traveling salesman problem, solutions are generated by sampling the set of possible permutations of the cities visited in a tour (Shi & Ólafsson, 2000a). In the case of AHA, the promising region is a hyperbox, and solutions are uniformly sampled in the promising region by discarding infeasible solutions. COMPASS samples solutions by iteratively applying minor alterations (moves) to an existing solution. This is done by randomly sampling a direction and performing a random-length step in the chosen direction while ensuring that the resulting point remains in the promising region.

Another important characteristic of simulation-based optimization is that simulators are oftentimes very computationally expensive to run. This implies that the number of solution evaluations that may be realistically carried out is limited. Despite this, there is a lack of research aiming to increase the small sample performance of DSO algorithms, i.e., increase the performance when a limited number of simulation calls is available to the algorithm. In this paper, we aim to address this gap by leveraging the mathematical formulation of the feasible space. This is a source of information that, to our knowledge, has not been explicitly used to guide DSO algorithms. We propose partitioning and sampling strategies that are directly informed by the problem's analytical properties, yielding a structurally-aware search that can allocate its limited simulation budget more effectively.

3. Nested clusters

In this section, we discuss NC, an enhancement of NP for problem (1)–(3) that integrates the proposed partitioning and sampling strategies. NP is well suited for this enhancement because its modular structure allows the partitioning and sampling steps to be modified directly. Algorithm 1 presents the pseudocode of NC: lines highlighted in red represent the steps where it differs from NP.

Let Φ^i be the promising region at iteration i. We initialize the algorithm by setting the initial promising region, Φ^0, to the complete search space $\Theta = \{x \in \{0,1\}^n : Ax \geq b\}$ (Line 2), and terminate the search when either the simulation budget (i.e., a maximum number of simulation evaluations) or the global computation time budget is exhausted. At each iteration i, we sample a set of n^c solutions, S^i, from Φ^i (Line 5). Sampling is performed by generating and solving an auxiliary problem based on the 0–1 linear programming formulation of the feasible space and using a randomized linear objective function. We use S^i to induce a partition of Φ^i into K disjoint sub-regions. To improve the quality of the produced clusters, we first remap the sampled solutions onto an alternative space by computing a set of features for each solution (Line 6). These features are linear combinations of variables which may be handcrafted based on expert knowledge of the

problem domain or be learned automatically. Computing the features of a solution can be seen as performing a linear projection of the problem variables, and the features are designed such that the projected space better reflects the aspects of the solutions that are most relevant to the simulated objective function. Partitioning is then performed by clustering the sampled solutions S^i (Line 7). We denote by Φ^i_k the kth sub-region of Φ^i, and by $\nu(\Phi)$ the super-region of Φ, i.e., Φ^i is the super-region of Φ^i_k.

For each sub-region Φ^i_k, with $k \in 1, \dots, K$, we sample a set of n^e solutions, $\hat{S}^i_k$, and evaluate them via simulation (lines 10–11). Additionally, at each iteration, if $\Phi^i \subset \Theta$, i.e., Φ^i is a strict subset of the feasible space, we also sample and evaluate a set of n^e solutions, $\hat{S}^i_0$, which are sampled outside of the current promising region (lines 14–15). We then compare the solutions in $\hat{S}^i_0$, to the solutions sampled from each sub-region of Φ^i, $\hat{S}^i_k$; if any solution in $\hat{S}^i_0$ achieves a better objective value compared to those in $\hat{S}^i_k$, $\forall\, k \in 1, \dots, K$, we backtrack the search (lines 16–18). Backtracking is performed by setting the promising region to its super-region, i.e., $\nu(\Phi^i)$. Otherwise, based on the evaluation of the solutions $\hat{S}^i_k$, we set the promising region for the next iteration, Φ^{i+1}, to the sub-region of Φ^i containing the best evaluated solution (Line 22); we then iterate. In essence, NC is an implementation of NP that uses a cluster-based strategy to partition Φ and a 0–1 linear programming-based strategy that is tailored to the DSO problem at hand to sample new solutions during the algorithm execution.

The remainder of this section is organized as follows. In Section 3.1 we present our cluster-based partitioning strategy. In Section 3.2 we introduce the concept of solution features and discuss how the use of features improves the proposed partitioning strategy. In Section 3.3 we discuss the proposed sampling strategy. In Section 3.4 we expand on the aforementioned sampling strategy by introducing an additive bias and discuss possible tuning options. Finally, in Section 3.5 we propose a greedy heuristic to generate high-quality solutions using the biasing strategy presented in Section 3.4.

Algorithm 1: The NC algorithm, lines highlighted in red differ from NP.

```
1  Algorithm Nested_Clusters()
2      Φ^0 ← Θ;
3      i = 0;
4      while not termination condition do
5          S^i ← Sample n^c random solutions in Φ^i by solving
             Formulation (4), (2), (3), (5) ;        // see Section 3.3
6          S̃^i ← Compute the features of each solution in S^i ;   // see
             Section 3.2
7          Cluster S̃^i using k-means ;              // see Section 3.1
8          {Φ^i_k, ∀ k ∈ 1, ..., K} ← Partition Φ^i based on computed clusters;
9          for k ∈ 1, ..., K do
10             Ŝ^i_k ← Sample n^e random solutions in Φ^i_k by solving
                 Formulation (4), (2), (3), (6) ;
11             s^i_k = min_{x∈Ŝ^i_k}(f̂(x));
12         end
13         if Φ^i ≠ Θ then
14             Ŝ^i_0 ← Sample n^e random solutions not in Φ^i by repeatedly
                 solving Formulation (4), (2), (3) until n^e solutions not in
                 Φ^i are found ;
15             s^i_0 = min_{x∈Ŝ^i_0}(f̂(x));
16             if s^i_0 < min_{k∈1,...,K}(s^i_k) then
17                 Φ^{i+1} ← ν(Φ^i);
18                 i ← i + 1;
19                 continue;
20             end
21         end
22         k̄ ← arg min_{k∈1,...,K}(s^i_k);
23         Φ^{i+1} ← Φ^i_k̄;
24         i ← i + 1;
25     end
26     return best evaluated solution
```

3.1. Partitioning strategy

To partition the feasible space, we use a cluster-based strategy. For simplicity, we initially present the proposed strategy without considering solution features. We will elaborate on how the procedure is expanded to incorporate features in Section 3.2.

At each iteration of NC, we partition the promising region (i.e., Φ^i). We first generate a set of n^c randomly sampled solutions in Φ^i, S^i (see Section 3.3). We then use a k-means algorithm to cluster the solutions in S^i with the number of clusters set to K.

The k-means algorithm operates by identifying a set of points (centroids) in $\mathbb{R}^n$, where n is the number of decision variables, each associated with a single cluster. We denote by $\mu^i_k \in \mathbb{R}^n$ the kth centroid in iteration i. The k-means algorithm assigns an unclassified solution $x' \in \Theta$ to the cluster k that minimizes the Euclidean distance between x' and μ^i_k.

Formally, x' is assigned to the cluster $k = \arg\min_{k' \in 1,\dots,K} |x' - \mu^i_{k'}|$. Therefore, solution x' is assigned to cluster k if $|x' - \mu^i_k| \leq |x' - \mu^i_{k'}|$, $\forall\, k' \in 1, \dots, K \;:\; k' \neq k$. When a feasible solution is equidistant from multiple centroids, we break ties lexicographically according to the cluster index. The previous inequality can be rewritten as: $\lambda^{\intercal}_{kk'} \cdot x' \geq \sigma_{kk'}$, $\forall\, k' \in 1, \dots, K \;:\; k' \neq k$, where the n-dimensional vector $\lambda_{kk'}$ and the scalar $\sigma_{kk'}$ define a hyperplane that is orthogonal to the segment that connects μ^i_k and $\mu^i_{k'}$ and remains equidistant from both points. Formally, $\lambda_{kk'} = \mu^i_k - \mu^i_{k'}$, and $\sigma_{kk'} = \lambda^{\intercal}_{kk'} \cdot \frac{\mu^i_k + \mu^i_{k'}}{2}$. The region of the feasible space associated with cluster Φ^i_k is therefore expressed through $K-1$ linear inequalities as $\Phi^i_k = \{x \in \Phi^i \;:\; \lambda^{\intercal}_{kk'} \cdot x \geq \sigma_{kk'} \quad k' \in 1, \dots, K \;:\; k' \neq k\}$.

Clusters $\{\Phi^i_k \;\forall\, k \in 1, \dots, K\}$ identify a partition of Φ^i. We note that the proposed partitioning strategy bears similarities to the one used by Hong and Nelson (2006), as both partition the feasible space using separating hyperplanes. The main difference between the two strategies is that Hong and Nelson (2006) do not perform a preliminary clustering step. Instead, they partition directly based on the set of evaluated solutions. This distinction is significant for partitioning spaces with binary decision variables, where many solutions are equidistant from each other. If a separating hyperplane is constructed directly between two feasible solutions, several other solutions may lie exactly on the hyperplane, leading to ambiguous region assignments or tie-breaking artifacts. In our approach we construct separating hyperplanes using clustering centroids, whose coordinates are typically fractional, producing hyperplanes that are less likely to pass through feasible solutions. Additionally, in the next section, we further expand on the proposed partitioning strategy by incorporating solution features, which allows us to better capture the structure of the feasible space.

3.2. Solution features

The clustering procedure described in Section 3.1 groups solutions based on their similarity in the decision variable space, using Euclidean distance as a similarity metric. This approach implicitly assumes that all decision variables are independent and equally important, as any difference between variables contributes equally to the distance. However, this assumption may be limiting; there might exist some relationships between variables that, if accounted for, could lead to a clustering that better captures the behavior of the simulator, i.e., achieving clusters that are more homogeneous in terms of simulated objective function values. In location problems, for example, solutions with similar spatial distributions of selected facilities may induce similar simulator responses even when the selected facilities do not match exactly.

To address this, we propose a general method for incorporating problem-specific knowledge into the clustering process by projecting solutions onto an auxiliary feature space prior to clustering. This projection only affects the clustering step and does not modify the feasible region searched by the algorithm. These features are defined as linear combinations of the decision variables that encode relevant relationships between variables. In this work, we assume that such relationships are available from domain knowledge and can be encoded as a linear projection. When such information is not available, the identity projection introduced below provides a generic fallback. The data-driven construction of feature projections represents an interesting direction for future research, but remains outside the scope of this paper. Additional examples of problem-specific feature projections are provided in Section 2 of the supplementary material.

Formally, let Q be an $m \times n$ feature matrix, where n is the number of decision variables and m is the number of features. Each row of Q defines one feature as a linear combination of the decision variables. Given a solution x, its feature space projection is $\bar{x} = Q \cdot x$. When $Q = I$, the identity matrix, clustering is performed directly in the original space. This identity projection does not require problem-specific information, but it may provide limited benefit when relevant similarities among solutions are better captured by aggregate features.

In the proposed framework, the feature matrix is used to project the solutions into the feature space prior to clustering. Thus, when partitioning $\boldsymbol{\Phi}^i$, each solution in S^i is projected into the feature space, and k-means clustering is performed based on these projected solutions. This results in centroids $\bar{\mu}^i_k$, for all $k \in 1, \dots, K$, with coordinates in the feature space. A given solution x is associated with cluster k if its feature projection $\bar{x}$ satisfies $\bar{\lambda}^\intercal_{kk'} \cdot \bar{x} \geq \bar{\sigma}_{kk'}, \forall\ k' \in 1, \dots, K\ :\ k' \neq k$. Here, $\bar{\lambda}_{kk'}$ and $\bar{\sigma}_{kk'}$ are computed as detailed in Section 3.1.

We note that $(\bar{\lambda}^\intercal_{kk'} \cdot Q)x = \bar{\lambda}^\intercal_{kk'}(Q \cdot x) = \bar{\lambda}^\intercal_{kk'} \cdot \bar{x} \geq \bar{\sigma}_{kk'}, \forall\ k' \in 1, \dots, K : k' \neq k$. Thus, when using solution features, cluster $\boldsymbol{\Phi}^i_k$ is expressed as $\boldsymbol{\Phi}^i_k = \{x \in \boldsymbol{\Phi}^i : (\bar{\lambda}^\intercal_{kk'} \cdot Q)x \geq \bar{\sigma}_{kk'}, \forall\ k' \in 1, \dots, K : k' \neq k\}$.

3.3. Sampling strategy

To sample solutions from the promising region $\boldsymbol{\Phi}^i$ we use an auxiliary 0–1 linear programming formulation, based on (1)–(3). Since the objective function (1) is evaluated via simulation and cannot be used directly in optimization, we replace it with a randomized linear combination of the decision variables:

$$\min\ \tilde{c}^\intercal x \tag{4}$$

(2), (3)

$$x \in \boldsymbol{\Phi}^i. \tag{5}$$

Here, $\tilde{c} \in \mathbb{R}^n$ is a random vector uniformly sampled from the unit sphere centered at the origin. Constraint (5) imposes that the sampled solution is inside the promising region $\boldsymbol{\Phi}^i$. Since $\boldsymbol{\Phi}^i$ is defined using linear constraints, formulation (4)–(5) remains a valid 0–1 linear program.

Solving formulation (4), (2), (3), (5) effectively samples a random vertex of the convex hull of $\boldsymbol{\Phi}^i$. Given that all variables are binary, all feasible solutions in $\boldsymbol{\Phi}^i$ lie at vertices of its convex hull. Thus, for any feasible solution in the current promising region $x \in \boldsymbol{\Phi}^i$, there exists a value of $\tilde{c} \in \mathbb{R}^n$, such that x is the optimal solution of formulation (4), (2), (3), (5). However, not all solutions have the same probability of being sampled using this method; i.e., the sampling is not uniform. More precisely, a feasible solution is sampled when the random vector $\tilde{c}$ belongs to the corresponding normal cone of the convex hull of $\boldsymbol{\Phi}^i$ at that solution. These normal cones generally have different solid angles; therefore, feasible solutions associated with larger normal cones are sampled with higher probability. This geometry-induced distribution need not be aligned with solution quality: a promising solution whose normal cone has a small solid angle may therefore be sampled infrequently. Nevertheless, every feasible solution retains a non-zero probability of being sampled. In the next section, we introduce a biasing mechanism that deliberately favors promising solutions, while still ensuring that every feasible solution retains a non-zero probability of being sampled.

To sample multiple solutions from $\boldsymbol{\Phi}^i$ using this method, we repeatedly solve formulation (4), (2), (3), (5), randomly generating a different vector $\tilde{c}$ for each solution we intend to sample. Furthermore, to sample solutions specifically from a sub-region of the current promising region $\boldsymbol{\Phi}^i_k$, we replace constraint (5) with the constraint:

$$x \in \boldsymbol{\Phi}^i_k. \tag{6}$$

This sampling strategy is appealing for its simplicity and generality: each sampled solution is obtained by solving a small-scale 0–1 linear programming formulation, whose computational cost is negligible relative to that of an expensive simulator, and the method requires no specialized tuning or assumptions about the geometry of the feasible space, making it robust for a variety of binary combinatorial problems.

3.4. Sampling bias

To boost the performance of NC, we consider biasing the sampling strategy to increase the likelihood of sampling high-quality solutions. We recall that the sampling strategy introduced in Section 3.3 is already non-uniform; however, this non-uniformity is induced by the geometry of the feasible region and is not a tunable component of the algorithm. The biasing mechanism introduced in this section complements this geometry-induced non-uniformity by using information collected during the search. This is done by manipulating the randomized objective function (4). Specifically, we introduce an additive biasing term γ to the random cost vector $\tilde{c}$, resulting in the following modified formulation:

$$\min\ (\tilde{c} + \gamma)^\intercal x \tag{7}$$

(2), (3), (6).

The core idea is to define γ such that $\gamma^\intercal x$ serves as a suitable proxy for the simulation-based objective function (1), increasing the likelihood of sampling more promising solutions. In doing so, we ensure that all feasible solutions retain a non-zero probability of being sampled, so that the global convergence guarantees inherited from NP still hold. The sampling strategy described in Section 3.3 can sample any feasible solution provided the cone generated by the randomized objective coefficient ($\tilde{c}$) spans the entire space $\mathbb{R}^n$. Therefore, since $\|\tilde{c}\| = 1$ (as it is sampled from the unit sphere), this condition is maintained if the magnitude of the biasing term γ is strictly less than one, i.e., $\|\gamma\| < 1$. Fig. 1(a) illustrates why this is the case in an unconstrained 2-dimensional space. The solution selected by solving the formulation depends on the quadrant in $\mathbb{R}^2$, where the vector $(\tilde{c} + \gamma)$ lies. We distinguish three cases. If $\|\gamma\| = 0$ (Fig. 1b) the objective vector $(\tilde{c} + \gamma)$ equals $\tilde{c}$, a random point sampled on the unit circle centered at the origin, which may belong to any quadrant. If $0 < \|\gamma\| < 1$ (Fig. 1c), $(\tilde{c} + \gamma)$ corresponds to a random point on the unit circle centered at γ. Since the circle contains the origin, $(\tilde{c} + \gamma)$ may belong to any quadrant, however it will be "biased" towards γ, which, in the example, would result in solution (1,0) being more likely to be sampled. Finally, if $\|\gamma\| \geq 1$ (Fig. 1d), the unit circle centered at γ no longer contains the origin. Thus, $(\tilde{c} + \gamma)$ cannot belong to some quadrants, meaning that some solutions (e.g., solution (0,1) in the displayed example) would have zero probability of being sampled, violating the conditions for global convergence.

To compute a suitable value for γ, we tune the simulator to rapidly evaluate solutions at low-fidelity. This may be done by (1) reducing the number of simulation scenarios considered, (2) reducing the duration of the scenarios, or (3) simplifying the simulator's logic, for example, by deactivating some of its capabilities. Strategies (2) and (3) might yield simulation results not directly comparable to the full-fidelity evaluations. This is acceptable as this low-fidelity model is exclusively used to compute the bias term γ. The precision of this low-fidelity

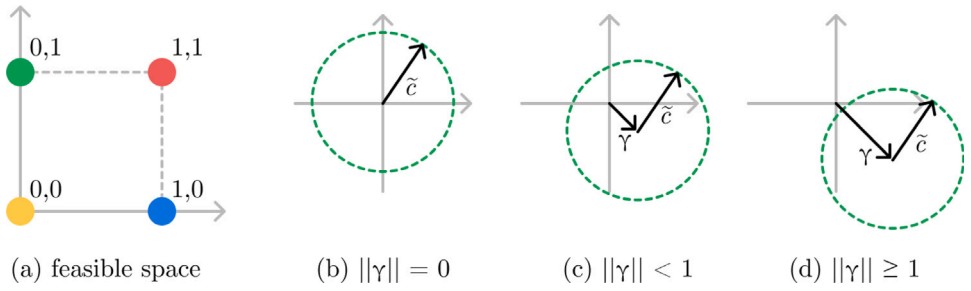


Fig. 1. Explanation of the impact of $\|\gamma\|$.

model does not need to be perfect; it only needs to provide a reasonably good directional signal towards promising areas of the search space.

The computation of γ is performed as a dedicated pre-processing step, for which we allocate a small portion of the total available computational time. During this phase, we randomly sample a large number of solutions and evaluate them using a low-fidelity simulator. We then fit a Ridge regressor (Hoerl & Kennard, 1970) to predict the low-fidelity objective values based on the sampled solution vectors x. Finally, the bias vector γ is computed as $\gamma = \frac{\alpha}{\|\theta\|}\theta$, where α is a scalar parameter controlling the strength of the bias, with $\alpha \in [0, 1)$. This ensures $\|\gamma\| = \alpha < 1$. In the process, regularization is used to avoid overfitting and to mitigate numerical precision errors. Ridge regression was preferred over LASSO (Tibshirani, 1996), as it retains all variables and avoids excluding features that may still carry valuable directional information. Additionally, we note that we assume that we are solving a minimization problem; when this is not the case, the sign of the bias should be inverted.

It is important to highlight that biasing term γ does not force the algorithm to re-select specific points identified during the low-fidelity training phase. Rather, it skews the sampling probabilities, making it more likely to sample solutions that are "similar" (in the sense captured by the linear model $\theta^\intercal x$) to previously observed good solutions. The randomization from $\tilde{c}$ ensures that a wide variety of solutions will still be generated. Even with a strong bias (i.e., α close to 1), the sampled solutions are generally different from those seen during training, and crucially, all feasible solutions retain a non-zero probability of being sampled. In other words, the bias shifts the search focus, it does not deterministically pick solutions.

3.5. Strong sampling bias

As noted above, the general biasing approach described in Section 3.4 still samples solutions from the promising region Φ^i stochastically. However, the learned bias term γ can also be leveraged in a more direct, deterministic manner. Given that $\gamma^\intercal x$ is constructed as a proxy for the true objective function (1), it is of interest to identify the solution that optimizes this proxy within a given sub-region Φ^i_k. Such a solution may be easily obtained by solving:

$$\min\ \gamma^\intercal x \tag{8}$$

(2), (3), (6).

However, only one such solution from each sub-region of Φ^i may be sampled using this method since the objective function (8) is not randomized. We boost the performance of NC by adding this greedy solution to the set of randomly sampled solutions that would be used to evaluate the quality of a sub-region. This has the advantage of identifying high-quality solutions at a low cost during the search. Additionally, this strategy also guarantees that at each iteration at least one solution strongly favored by the proxy $\gamma^\intercal x$ is sampled from each sub-region Φ^i_k.

While this does not affect the algorithm's convergence properties (see Section 3.6), the lack of such a guarantee can reduce search consistency in small-sample scenarios, as a promising sub-region might be discarded due to non-representative random sampling.

3.6. Global convergence

The Nested Partitions (NP) framework is known to converge to a global optimum in finite time under a set of mild conditions (Shi & Ólafsson, 2000b). This guarantee is derived from modeling the algorithm's progression as a finite-state Markov chain where the only absorbing states are the global optima. For combinatorial optimization, the convergence holds provided that:

1. The partitioning scheme is complete, meaning it can recursively refine disjoint sub-regions until any single solution can be isolated in a singleton region.
2. The sampling procedure is exhaustive, assigning a strictly positive probability of selection to every feasible point within the region being sampled.

The NC algorithm is designed to preserve these structural properties.

1. The partitioning strategy is valid as it generates disjoint sub-regions that fully cover the current promising region. This process guarantees that singleton regions can be reached after a finite number of refinements, provided that the feature mapping is injective on the feasible set, i.e., no two distinct feasible solutions map to the same feature vector. In practice, this can be guaranteed by appending the (scaled) original variables to the problem-specific features, which ensures injectivity while preserving the intended clustering behavior.
2. The sampling strategy is exhaustive as all feasible solutions have a strictly positive probability of being sampled as discussed in Sections 3.3 and 3.4.

Since NC satisfies these core requirements for partitioning and sampling, it inherits the asymptotic global convergence guarantee of the NP framework, as all other components of NC are identical to the classical implementation of NP and NC does not introduce any mechanisms that would break these conditions.

4. A case study on electric vehicle charger location for ride-hailing services

To test our approach, we apply it to the problem of locating charging and parking infrastructure to support the operation of a private fleet of electric ride-hailing vehicles and maximize its profit. We refer to this problem as the Electric Vehicle Charger Location Problem for Ride-hailing Services (EVCLPR). For a fixed fleet, the problem consists of locating battery chargers and parking spots that vehicles use to recharge or reposition in response to anticipated demand. To measure

the objective, we solve the operational level problem of dynamically assigning vehicles to requests as they arise, and managing the fleet by recharging and repositioning the vehicles. The fleet management component in our EVCLPR, can be seen as a generalization of the ride-hailing problem introduced by Kullman et al. (2022). Contrary to their setting, we consider that charging stations have limited capacity in terms of the number of vehicles that may be simultaneously charged. On the other hand, similar to their setting, we assume that the problem objective is to maximize the long term expected profit (i.e., the revenue obtained by serving requests, minus the traveling and charging cost). A detailed discussion of the problem definition of the EVCLPR is available in the online version of this paper. The EVCLPR emerged as a good testbed for our approach for three main reasons: (i) the availability of a well-validated ride-hailing simulator that can be adapted to evaluate EVCLPR solutions, (ii) its combinatorial nature, and (iii) its scientific, practical, and societal relevance.

A crucial component of any simulation-based optimization approach is the simulator used to evaluate solutions or constraints. Designing and implementing an efficient and effective simulator can be a daunting task. We leverage the simulator developed by Kullman et al. (2022), based on the publicly available Pyhail package (Kullman, 2022). The latter was developed for the Dynamic Ride Hailing stream of the 12th DIMACS Implementation Challenge (DIMACS, 2022), where it proved to be an effective tool, well-validated, and based on high-quality data. We adapt this simulator to the EVCLPR setting by incorporating capacitated charging stations, FIFO charging queues, reduced battery capacity, and the infrastructure-location decisions considered in this paper. The simulator used for these experiments has been made open source and is available as part of OpenHail (Schettini et al., 2026). Section 1 of the supplementary material reports the formulation of the EVCLPR and the main modeling details of the simulator used to evaluate candidate solutions.

Charger location problems are becoming increasingly relevant with the growing adoption of electric vehicles (EVs). Naturally, the operations research community has been very active in this area. For brevity, we do not discuss the latest advances in this emerging stream of research here, but we provide a brief yet thorough literature review in Appendix B. It is worth noting, however, that several authors have used simulation-based approaches to evaluate or optimize the location of charging infrastructure (Bi et al., 2017; Márquez-Fernández et al., 2019; Pagani et al., 2019; Sweda & Klabjan, 2011; Viswanathan et al., 2016; Wolbertus et al., 2021). Moreover, several of these studies have been carried out in the context of planning charging infrastructure for fleets of electric taxis (Anastasiadis et al., 2021; Pan et al., 2019; Yang et al., 2023; Zhang et al., 2020). Nonetheless, to the best of our knowledge, no contribution in the simulation-based optimization literature has tackled the challenge of simultaneously locating both charging and parking facilities for a ride-hailing fleet, as we do in this paper.

5. Computational results

We conduct a set of computational experiments to evaluate the effectiveness of NC on the EVCLPR. To complement this primary case study, Section 2 of the supplementary material reports controlled synthetic tests on four additional binary simulation–optimization problems with different combinatorial structures and stochastic evaluators. In Section 1 of the supplementary material, available in the online version of this paper, we report the formulation of the ride-hailing problem and the main modeling adaptations introduced in the simulator used to evaluate candidate solutions. The simulation parameters are taken from Kullman et al. (2022). Thus, all experiments are based on data from the island of Manhattan, and ride-hailing requests are generated from historical data provided by the taxi and limousine commission (Schneider, 2019). Compared to Kullman et al. (2022), we reduce the EV's battery capacity to 20 kWh to represent a more constrained operating scenario. To evaluate an infrastructure configuration, we run the simulator considering a set of 5 randomized scenarios, each simulating demand for three consecutive workdays. We selected this simulator configuration empirically to balance the accuracy of the estimation of $f(x)$ with computational time.

Table 1
Summary of the tested algorithms.

	No cluster-based partitioning	Cluster-based partitioning
No biased sampling	NP	NC
Biased sampling	NP-B	NC-B
Biased sampling with strong bias	NP-SB	NC-SB

We present the results of our algorithm on several configurations with respect to the number of vehicles operating in the fleet and the number and capacity of the charging stations to be placed. For convenience, in Table A.3, found in Appendix A, we report a summary of the parameters used in the experiments.

The simulator, as well as all the tested algorithms, are implemented using Python version 3.10 and GUROBI (Gurobi Optimization, LLC, 2023). We conduct all experiments on the Narval general-purpose cluster (Digital Research Alliance of Canada), setting up each compute node with a single thread of an AMD Rome 7532 processor clocked at 2.40 GHz, with 10 GB of available memory. The evaluation of a solution requires between 9 and 14 min, depending on the solution being simulated, and the number of vehicles operating in the system. Each run of each algorithm is allocated exactly three days of computational time; this accounts for the entire runtime, including sampling, partitioning, and evaluating the solutions.

As a practical measure to improve search diversification, we implemented a reset procedure in NP and NC which is triggered if the promising region converged to a singleton. In such cases, the region was expanded back to the entire feasible space, allowing the search to continue until the computational time limit was reached.

When this happens, the best solution found corresponds to the best solution found across all resets of the algorithm.

To evaluate the impact of all proposed algorithmic components, we implemented several versions of both NP and NC. For NP, we evaluate a standard implementation (NP), an additional implementation using the biased sampling mechanism described in Section 3.4 (NP-B), and an implementation using biased sampling plus strong bias to sample the first solution of each sub-region as described in Section 3.5 (NP-SB). For NC, we evaluate the standard implementation (NC) and two implementations with biased sampling (NC-B) and strong bias (NC-SB). We refer to Table 1 for a summary of the tested algorithms.

For each instance, we run each algorithm five times with different random seeds. Note that we test all algorithms on the same set of seeds. For all versions of NC, we set the number of clusters used in k-means to $K = 4$, and the number of samples per cluster to $n^e = 5$. The set of features used in NC (see Section 3.2) is obtained by overlaying a 10-by-30 grid on the map of Manhattan and counting the number of charging stations installed in each cell of the grid. For a given solution x, each feature corresponds to a grid cell, and its value is the sum of binary variables representing charging stations installed at candidate locations within that cell. These features are intended to capture the spatial distribution of the charging infrastructure, so that solutions with similar aggregate coverage patterns may be clustered together even when they select different individual candidate locations. In Fig. 2, we show the grid used to generate the problem features used in NC. To preserve the global convergence property of NC (see Section 3.6), we augment the feature space by appending the original decision variables to the problem-specific features. The original variables are scaled by a factor of 0.1 so that they remain available to ensure injectivity, but with reduced influence on the clustering decisions of NC.

For the biased versions of the tested algorithms we set the bias intensity to $\alpha = 0.9$. To compute the bias, we allocate three hours of

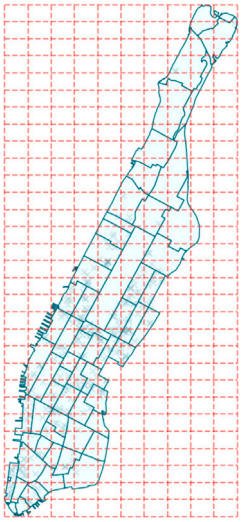


Fig. 2. The grid used to calculate the features.

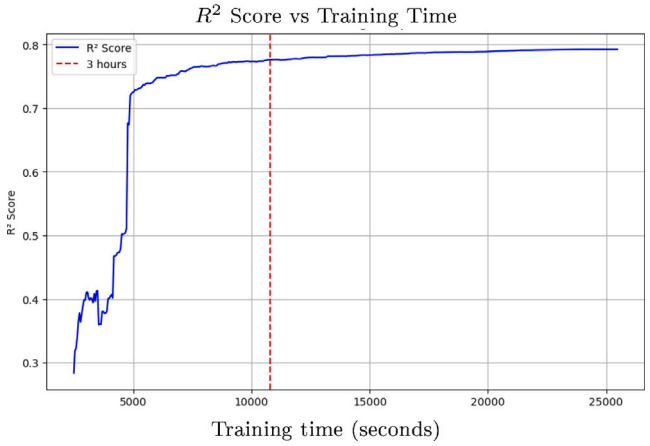


Fig. 3. Evolution of the R^2 score based on training time.

the total available computational time and we configure the simulator to consider a single scenario of a single workday. The three-hour time limit for sampling and evaluating solutions was determined experimentally. We cross-validate the predictive performance of the regressor as more data is collected. In Fig. 3 we plot the R-squared of the regressor as a function of training time. We found that after three hours of sampling (demarcated using a red dashed line in Fig. 3), the predictive quality of the regressor converged to a stable value, indicating that the regressor was able to capture the underlying signal of the low-fidelity simulator.

In Table 2, we report a summarized comparison of the results achieved by the tested algorithms. Each instance is identified using a string v-r, where v denotes the number of vehicles operating in the environment, and r denotes the number of charger locations. Rows identified by "all", report the results averaged across all instances for each algorithm. For each instance and each algorithm other than NP (i.e., NP-B, NP-SB, NC, NC-B, NC-SB), we report the gap between the solution found by the algorithm and the best solution found by NP, averaged across the five repetitions. Specifically, let ϵ be the set of seeds used in the evaluation, and let ζ be the set of instances; we denote by f_{is}^{NP*} the objective value of the solution found by NP for instance i using seed s. Similarly, we denote by f_{is}^{NX*} the objective value of the solution found by algorithm NX for instance i using seed s. The reported gap is computed as follows:

$$gap_i = \frac{\sum_{s\in\epsilon}(f_{is}^{NX*} - f_{is}^{NP*})}{\sum_{s\in\epsilon} f_{is}^{NP*}}.$$

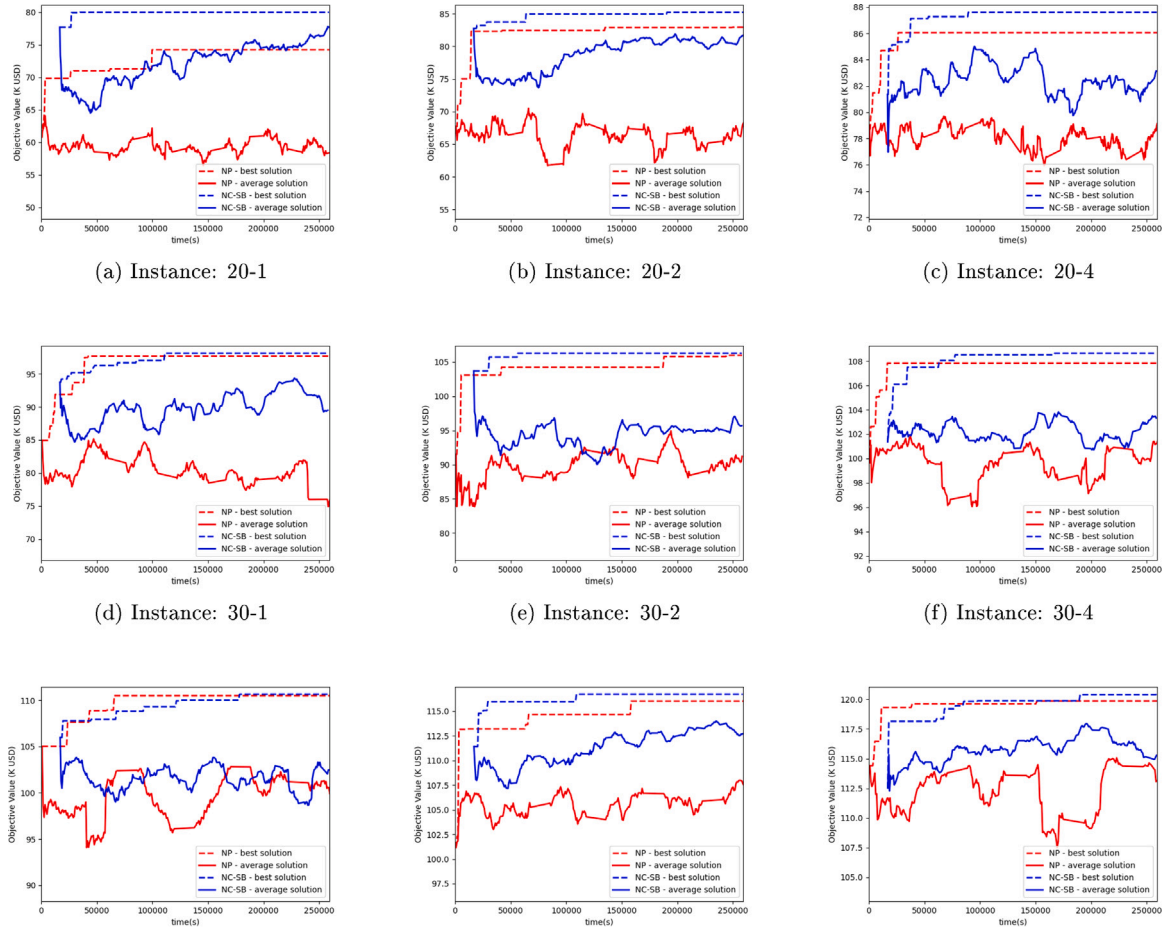
Additionally, for each instance and each algorithm, we report:

- the average objective value of the best solution found in each run, i.e., $\frac{\sum_{s\in\epsilon} f_{is}^{NX*}}{|\epsilon|}$,
- the average objective value of all solutions during the entire run, across all seeds; let P_{is}^{NX} be the set of solutions evaluated by algorithm NX, in instance i with seed s, this value is computed as $\sum_{s\in\epsilon} \frac{\sum_{p\in P_{is}^{NX}} f_{is}^{NXp}}{|P_{is}^{NX}|}$,
- the average number of times the simulator has been called to evaluate a solution (# iterations), and
- the average percentage of time spent not simulating solutions (i.e., sampling solutions, or partitioning the search-space).

Table 2 shows that using bias, and, especially, strong bias, leads to an improvement in the objective value of the best solutions found. More precisely, in the case of NP, using bias and strong bias leads to an average improvement of 0.9% (NP-B) and 1.4% (NP-SB). This comes at the expense of 3 h of CPU time (out of 72) that are devoted to tuning the bias, which translates into approximately 5% fewer evaluated solutions. A similar trend is observed in NC, where the improvement ranges from 0.4% to 6.6% when bias and strong bias are used. The impact of biased and strongly biased MILP-based sampling is also observed in the average objective value of the solutions. NP achieves an average objective of 87.6, a figure that increases to 91.2 when either bias (NP-B) or strong bias (NP-SB) is used. Similarly, NC reports an average objective of 87.9, but that value increases to 92.4 for NC-B and 94.2 for NC-SB. The data also show the value added by our clustering procedure. A quick look at the summary rows in columns 3 and 9 reveals that NC improves the average gap in the best solution found by 0.6 percentage points compared to NP. Similarly, NC-B improves it by 0.2 percentage points (1.1% vs. 0.9%) compared to NP-B, and NC-SB improves it by 0.8 percentage points compared to NP-SB (2.2% vs. 1.4%). MILP-based clustering also improves the average objective value of the solutions (columns 5 vs. 11). This is especially notable when comparing NC-SB (94.2) against NP-SB (91.2). In conclusion, this experiment showed that the compounded effects of our sampling and clustering procedures (NC-SB) result in an average improvement of 2.2% in the best solutions found by the standard implementation of NP.

The results in Table 2 also reveal that all tested algorithms spend nearly their whole computational budget running simulations. As columns 7 and 13 show the time spent sampling and partitioning-clustering is only a fraction (typically less than 0.5%) of the whole computational time.

Fig. 4 shows a head-to-head comparison between NP and NC-SB (i.e., the best-performing version of our algorithm). Each sub-figure shows the performance of the two algorithms on a single instance running with the same (random) seed. We observed similar performances running with other random seeds, but for brevity we do not report those results here. The x-axis of each sub-figure measures the algorithm progression (in seconds), while the y-axis reflects the objective value (in thousands of USD) of the solutions evaluated. The sub-figures report the performance of each algorithm using (i) a dashed line indicating the estimated objective function value of the best solution evaluated over the algorithm runtime and (ii) a solid line, where each point shows the rolling average of the last 20 solutions evaluated by each algorithm. In other words, the value of the solid line at x = 1000 corresponds to the average objective value of the last 20 solutions from the promising region that had been evaluated at time t = 1000 s. We chose the value of 20 to reflect the number of solutions that are evaluated at each iteration



(a) Instance: 20-1 (b) Instance: 20-2 (c) Instance: 20-4

(d) Instance: 30-1 (e) Instance: 30-2 (f) Instance: 30-4

(g) Instance: 40-1 (h) Instance: 40-2 (i) Instance: 40-4

Fig. 4. Comparison of evaluated solutions by NP and NC-SB. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

of the two algorithms. Note that the blue lines associated with NC-SB do not start at time 0, as the first 3 h of computation are spent training the bias. The data in the figures show that the average objective value of the solutions (i.e., solid line) evaluated by NC-SB is consistently higher than that of the solutions evaluated by NP. Additionally, we observe that the average objective value of the NP solutions remains relatively stable throughout the algorithm runtime. Conversely, on several instances, we observe a positive trend in the average objective value of NC-SB's solutions. This indicates that, throughout its execution, the algorithm identifies regions of high-quality solutions and concentrates the sampling there.

Fig. 5 presents a comparison between NP and NC-SB in terms of the cumulative distribution function (CDF) of the solutions they explore and evaluate. In each sub-figure, the x-axis reports the objective function, expressed in thousands of USD, while the y-axis measures the CDF. In particular, the solid line represents the percentage of solutions that achieve a specific objective value or better. For instance, if the curve of an algorithm reaches the 0.2 mark on the y-axis at a value of 100 thousand USD on the x-axis, it means that 20% of the sampled solutions have an objective value of 100 thousand USD or higher. In general, the more the curve of an algorithm is shifted to the right, the higher the probability it samples and evaluates good solutions. The shaded regions surrounding each curve represent the standard deviation of the CDF based on the five repetitions. From these graphs, we observe that the line associated with NC-SB is consistently higher and shifted to the right compared to the line associated with NP. Taking as an example instance 20-1, from the PDF, we observe that for NC-SB 50% of the sampled solutions achieve an objective above 72 thousand USD, while for NP less than 10% of the sampled solutions achieve the same objective. We observe similar trends for all instances. Hence, solutions evaluated by NC-SB are, on average, of higher quality when compared to the solutions evaluated by NP.

Fig. 6 zooms in on the behavior of NC-SB. It shows the CDF depending on the *depth* of the search; that is, the number of times the feasible space has been partitioned to obtain the promising region. At depth 1, the promising region corresponds to the entire feasible region, and therefore the first CDF describes randomly sampled solutions. Subsequent plots correspond to the CDF associated with gradually narrower regions. As mentioned before, CDFs that skew towards the right indicate higher proportions of better solutions. The results in Fig. 6 confirm that the sampling and partitioning (i.e., clustering) mechanisms embedded into NC-SB achieve their goal of identifying better-quality regions and focusing the search efforts there.

Table 2
Summary of results.

Instance	Algorithm	Gap to NP (%)	Best objective (K$)	Average objective (K$)	# iterations	Non-simulation time (%)	Algorithm	Gap to NP (%)	Best objective (K$)	Average objective (K$)	# iterations	Non-simulation time (%)
20-1			75.0	59.1	476.8	0.2		1.0	75.7	63.3	469.0	0.3
20-2			82.4	67.3	422.8	1.2		1.6	83.8	68.8	458.4	0.5
20-4			86.8	78.4	385.4	1.5		0.4	87.2	76.8	362.6	0.2
30-1			97.5	79.9	358.8	0.1		−0.5	97.0	82.5	349.4	0.2
30-2	NP		103.8	89.5	345.8	0.2	NC	0.1	103.9	89.2	341.6	0.3
30-4			108.3	99.3	330.6	0.2		0.1	108.4	97.2	327.8	1.3
40-1			110.0	98.3	311.6	0.2		1.5	111.7	99.4	302.4	0.4
40-2			115.5	104.4	316.6	0.2		0.9	116.5	104.1	310.0	0.4
40-4			120.4	112.2	302.4	0.1		0.7	121.2	109.3	298.6	0.9
All			**100.0**	**87.6**	**361.2**	**0.4**		**0.6**	**100.6**	**87.9**	**357.8**	**0.5**
20-1		3.0	77.3	63.8	456.4	0.2		1.2	76.0	65.3	449.4	0.6
20-2		1.9	84.0	73.5	431.6	0.1		2.2	84.2	75.3	439.2	1.0
20-4		0.4	87.2	80.8	378.2	0.4		1.1	87.7	82.1	413.2	0.5
30-1		0.2	97.7	84.0	294.6	1.4		0.4	97.9	84.6	336.0	0.5
30-2	NP-B	1.3	105.1	94.5	331.4	0.1	NC-B	1.4	105.2	94.3	320.8	0.5
30-4		0.1	108.4	102.0	301.8	0.1		0.6	109.0	103.3	286.0	3.4
40-1		0.5	110.6	100.0	277.4	0.2		1.2	111.4	101.8	284.4	0.6
40-2		0.6	116.2	107.4	235.8	0.2		0.8	116.4	108.4	286.2	0.6
40-4		0.4	121.0	114.4	276.4	0.2		0.7	121.3	116.5	269.2	0.6
All		**0.9**	**100.8**	**91.2**	**331.5**	**0.3**		**1.1**	**101.0**	**92.4**	**342.7**	**0.9**
20-1		6.6	80.0	64.0	395.0	0.3		6.6	80.0	70.3	471.8	0.6
20-2		2.1	84.1	73.5	435.8	0.1		3.4	85.2	78.5	447.6	0.9
20-4		0.6	87.3	80.7	366.6	1.3		0.8	87.5	82.5	406.6	0.4
30-1		0.7	98.2	83.9	292.0	0.4		1.5	99.0	87.7	356.0	0.5
30-2	NP-SB	1.3	105.1	94.9	290.8	0.3	NC-SB	2.6	106.5	98.0	361.4	1.4
30-4		−0.1	108.3	102.0	267.0	0.3		0.8	109.1	103.2	321.2	0.5
40-1		0.4	110.5	100.0	290.2	0.2		1.4	111.6	101.6	286.8	0.6
40-2		0.8	116.3	107.5	291.4	0.1		2.0	117.7	110.0	289.0	0.8
40-4		0.4	121.0	114.4	277.2	0.2		0.5	121.0	116.2	271.2	1.0
All		**1.4**	**101.2**	**91.2**	**322.9**	**0.4**		**2.2**	**102.0**	**94.2**	**356.8**	**0.7**

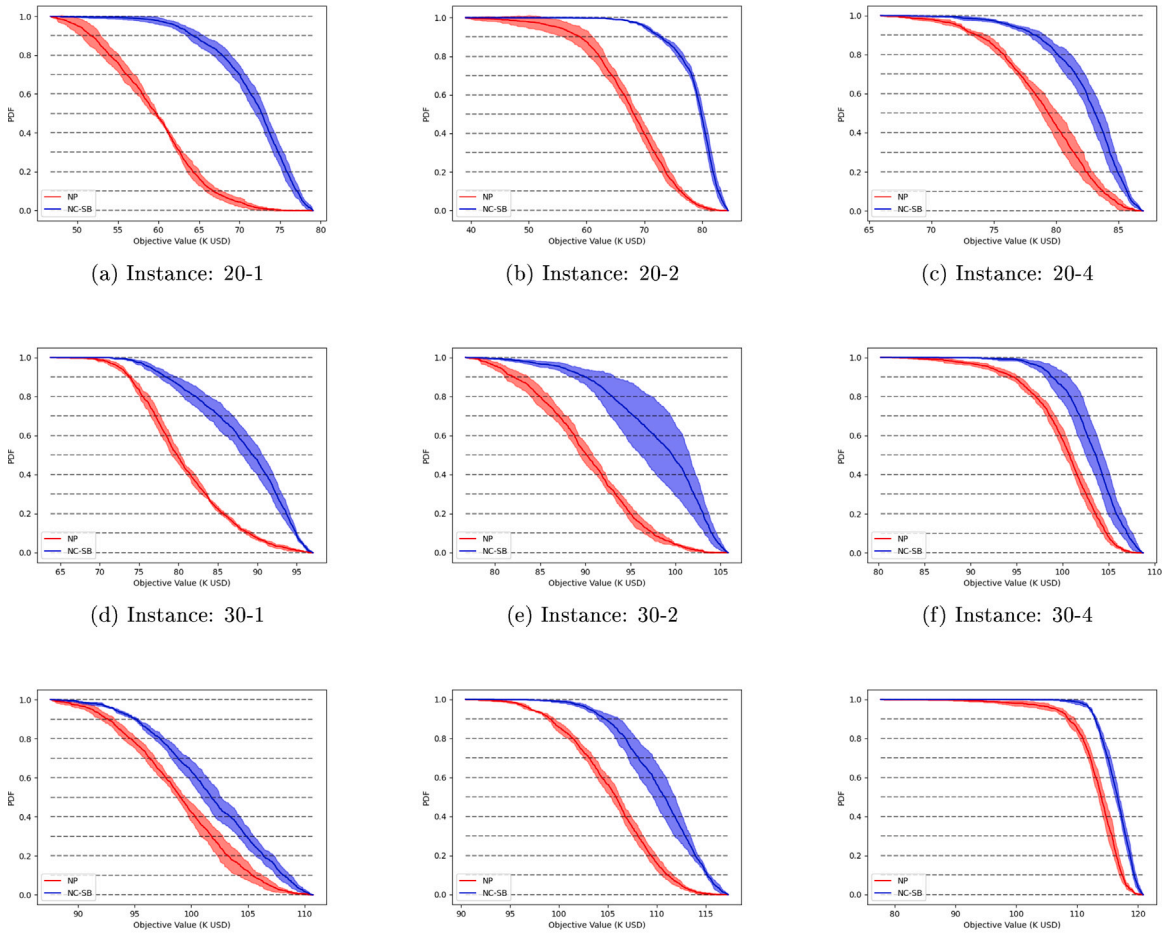
6. Conclusions

This paper proposes Nested Clusters (NC), an enhancement of Nested Partitions (NP) for binary DSO problems whose feasible space can be represented using a 0–1 linear programming formulation. The proposed partitioning strategy uses the k-means algorithm to identify partitions of the search space. Sampling is performed using an auxiliary formulation of the problem, based on the 0–1 linear programming formulation of the feasible space that is augmented using a randomized objective function. To strengthen the strategy, we devise a mechanism to incorporate problem-specific information into the framework through solution features. Additionally, we develop a method for biasing the sampling of solutions to favor high-quality regions of the search space. We implement the proposed strategies in the algorithm of NP, first testing the impact of each proposed component on the EVCLPR as our primary benchmark, and later conducting controlled synthetic tests on four additional problems with distinct structures and stochastic evaluators. In our computational experiments, the proposed sampling and clustering strategies improve the NP baseline on the tested EVCLPR instances; their compound effect yields an average improvement of 2.2% on the best solutions evaluated on the tested case study. Additionally, the proposed method leads to a significant 7% improvement in the average objective value of the solutions evaluated during the algorithm runtime. Through further analysis, we show that the proposed approach is able to achieve enhanced convergence speed and effectively focus on regions of high-quality solutions throughout its execution.

In future research, we intend to extend the partitioning and sampling strategies to other adaptive random search frameworks and to more general problem configurations, not limited to problems that may be represented using 0–1 linear programming formulations. We remark that NC may already integrate non-linear constraints in the problem definition through the simulator. Indeed, any DSO problem may be extended to include additional non-linear or stochastic constraints by evaluating them when a solution is simulated and heavily penalizing their violation in the simulated objective function. An interesting extension of this work involves using simplified mathematical programming models to approximate the problem structure and embed this information within the framework for guiding the search. This will extend ideas that have been successfully employed in continuous spaces (e.g., importance sampling, inverse cumulative distribution function sampling) to discrete spaces.

In this paper we assumed that the feature matrix Q, which is used to project solutions onto an auxiliary space prior to clustering, could be derived from problem-specific expert knowledge. However in some cases expert knowledge might be unavailable, incomplete, or difficult to formalize into a linear projection. For such cases, Q may be learned using data-driven techniques similar to the approach we used to train the bias term in Section 3.4. Techniques such as Partial Least Squares (PLS) regression (Fornell & Cha, 1994) and Locality Preserving Projections (LPP) (He & Niyogi, 2003) offer promising starting points. Preliminary experiments suggest that, when properly tuned, these learned features can outperform tailored ones, offering promising future research directions.

Another promising direction concerns the integration of sample-reuse mechanisms into NC. Recent work on product-space stochastic annealing (Ball et al., 2024) provides one relevant starting point, since it studies how simulation samples can be reused without distorting the acceptance logic of stochastic search. In particular, incremental evaluation could reduce the cost of screening candidate solutions within promising regions or training low-fidelity bias terms. Future research will investigate how sample reuse can be combined with the partitioning and sampling mechanisms proposed in this paper.



(a) Instance: 20-1 (b) Instance: 20-2 (c) Instance: 20-4

(d) Instance: 30-1 (e) Instance: 30-2 (f) Instance: 30-4

(g) Instance: 40-1 (h) Instance: 40-2 (i) Instance: 40-4

Fig. 5. Cumulative distribution function of evaluated solutions by NP and NC-SB.

CRediT authorship contribution statement

Tommaso Schettini: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Fausto Errico:** Writing – review & editing, Validation, Supervision, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Jorge E. Mendoza:** Writing – review & editing, Validation, Supervision, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization. **Carolina Osorio:** Writing – review & editing, Validation, Supervision, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors gratefully acknowledge financial support from *IVADO* through its Strategic Research Funding Program, Integrated Machine Learning and Optimization for Decision Making under Uncertainty, from the Natural Sciences and Engineering Research Council of Canada (NSERC/CRSNG) through Discovery Grant RGPIN-2022-03916, from *HEC Montréal* through the research professorship in Clean Transportation Analytics and from the *Scale AI Chair* in Urban Mobility and Logistics.

Appendix A. Simulation parameters

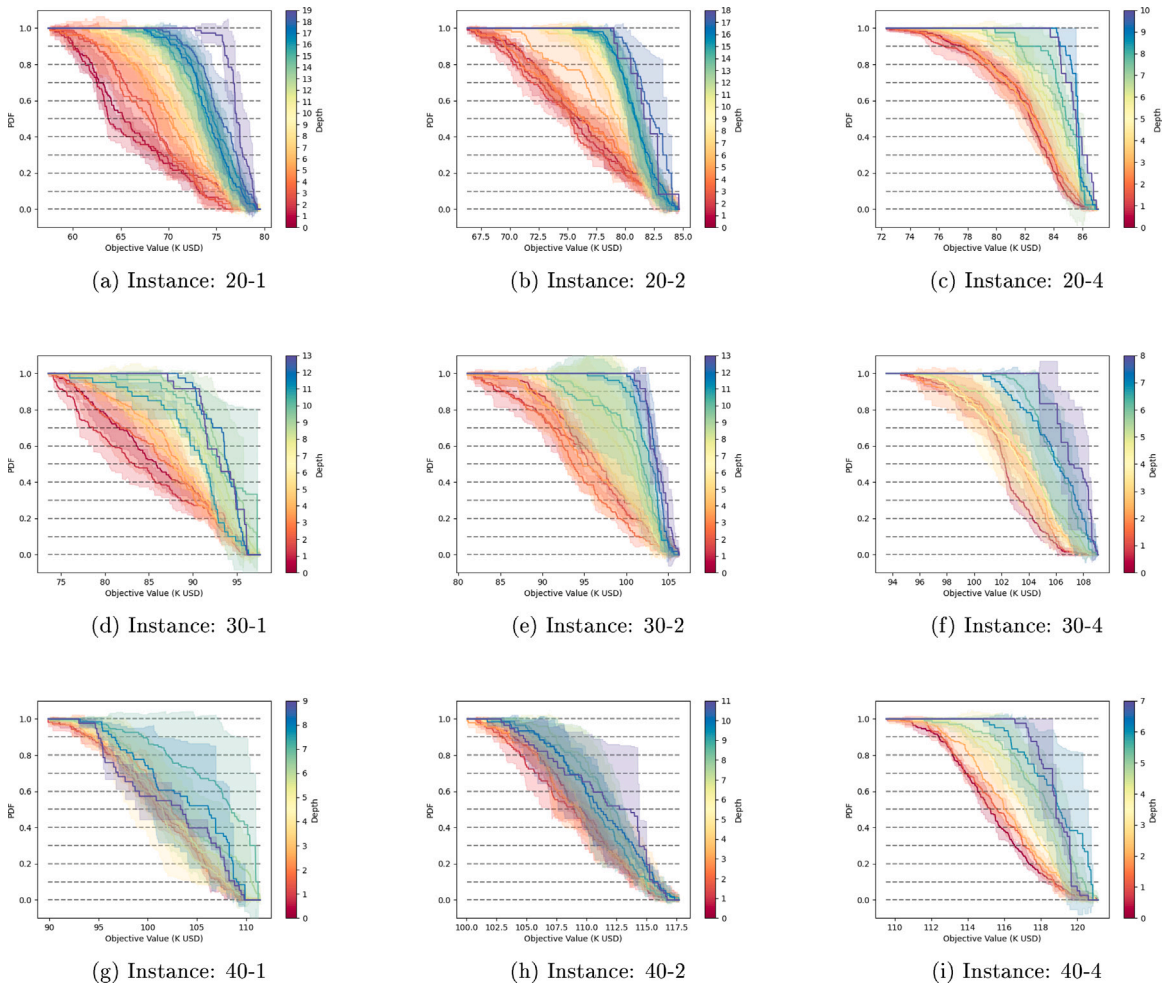
See Table A.3.

Appendix B. Literature review on charger location problems

Charger location problems have received considerable research interest in recent years. We refer the reader to Adenaw and Lienkamp (2020) and Kchaou-Boujelben (2021) for two recent reviews.

An important area of the literature has focused on flow-capturing models, which are designed to place charging stations in locations that effectively capture demand flows (Capar et al., 2013; Kuby & Lim, 2005, 2007; MirHassani & Ebrazi, 2013; Wang & Lin, 2009). Authors often extend the flow-capturing location model to include a limited tolerance for deviations in the captured flows (Arslan et al., 2019; Göpfert & Bock, 2019; Kim & Kuby, 2012; Yıldız et al., 2016). Several other authors have studied stochastic or robust versions of the problem (Boujelben & Gicquel, 2019; Hosseini et al., 2022; Lee & Han, 2017; Wu & Sioshansi, 2017). Another common extension of flow-capturing models has considered limited station capacity, often optimizing the sizing of the stations in addition to their locations (Wang et al., 2018; Yıldız et al., 2019).

A recent trend in the literature concerns the location of charging infrastructure for a fleet of taxis or, more generally, ride-hailing vehicles. Several authors consider data-driven approaches to determine the location of chargers based on the density and spatial distribution of historical demand data (Cilio & Babacan, 2021; Jia et al., 2017; Tu et al., 2016). Wang et al. (2021) proposes a heuristic approach for locating



(a) Instance: 20-1 (b) Instance: 20-2 (c) Instance: 20-4

(d) Instance: 30-1 (e) Instance: 30-2 (f) Instance: 30-4

(g) Instance: 40-1 (h) Instance: 40-2 (i) Instance: 40-4

Fig. 6. Cumulative distribution function of the evaluated solutions at various search depths for NC-SB.

Table A.3
Summary of the parameters used in the experiments.

Parameter	Value	Parameter	Value
Episode length	72 h	Max customer waiting time	5 min
Episode start time	3:00 am.	Number of requests	[2000] requests/day
CS charging rate	72 kW	Number of vehicles	[20, 30, 40] EVs
Vehicle battery capacity	20 kWh	Number of charger locations	[1, 2, 4]
Vehicle energy consumption	0.15 kWh/km	Number of parking locations	5
Vehicle speed	13 km/h	Number of chargers per station	[1, 2, 4]
Travel costs	$0.53/km	Number of candidate locations	302
Base fare	$10.75		
Per-distance fare	$4.02/km		

and sizing chargers in an urban network. The problem is tackled using a multi-step approach, where demand estimation, demand allocation, and location sizing are performed in three distinct steps. A reduction step is then applied to iteratively refine the solution by removing inefficient charging stations. Godbersen et al. (2024) presents a robust optimization approach for the planning of the charging infrastructure for a metropolitan taxi fleet. This is done using a two-stage optimization approach. In the first stage, the charging infrastructure is determined. In the second stage, the feasibility of the solution is evaluated.

Simulation-based models have seen wide use in the context of charger location problems, both to estimate demand data for the chargers and to evaluate proposed charger configurations. Several authors (Márquez-Fernández et al., 2019; Pagani et al., 2019; Sweda & Klabjan, 2011; Viswanathan et al., 2016) propose agent-based simulations to evaluate charging station layouts and estimate future adoption rates. Xi et al. (2013) employ simulation to estimate EV adoption rates and compute charger demand, and optimize the charging infrastructure to maximize the number of served EVs. Bi et al. (2017) develop an iterated simulation-based heuristic for locating chargers in a city network. Starting from a full network, the heuristic iteratively removes chargers that are deemed to be unnecessary by the simulation until a minimal configuration is reached. Zhuge and Shao (2018) consider a simulation- and agent-based optimization approach for locating and sizing charging infrastructure in a road network. Demand is estimated through simulation, whereas the location decisions are performed by individual agents placed at various points of the network that operate through a specialized rule-set over multiple iterations. Zang et al. (2018) use simulation to estimate EV demand, and subsequently use a multi-step approach for location and sizing of the charging infrastructure of a road network. Kong et al. (2019) present a two-step approach for location and sizing of fast charging stations on a road network. First, a set of preliminary candidate locations is heuristically determined using

a sequential space partitioning approach, then the result is validated and refined through simulation. Wolbertus et al. (2021) present a simulation- and agent-based optimization approach for locating and sizing charging infrastructure in a road network where the charging decisions are also performed by simulated agents. The simulation includes both the agent's choices to adopt an electric vehicle and the operator's choices of locating and sizing the chargers. The authors analyze the impact of different operator policies and usage of EVs. Pan et al. (2019) address the problem of siting chargers for a fleet of electric taxis. The evaluation of the solution is done through the simulation of the taxi system in an agent-based fashion. All taxis are individually simulated with objective functions. Network load is explicitly accounted for, and chargers use a queuing system in the optimization to represent the limited capacity of the infrastructure. Candidate sites are extracted from the input trajectory data, which is also used to estimate demand. The optimization is performed using a genetic algorithm. Anastasiadis et al. (2021) present a charger location approach for the placement of private chargers for a ride-hailing taxi service. The method uses a simulation-based approach to estimate the charger demand that is required in the service. The points where the agents had to go to charge are used as demand points for a charger location problem that aims to locate and size the chargers for the region. In the optimization, the model aims to minimize travel time to chargers, installation costs, and expected charger waiting time for the charger usage. To estimate the waiting time at the various chargers, a queuing model is used. Zhang et al. (2020) present a two-step simulation-based approach for locating and sizing chargers for a fleet of ride-hailing autonomous electric vehicles. The approach uses an agent-based simulation procedure to estimate the charging demand of the fleet. A k-means approach is then used to determine suitable charger locations and, finally, chargers are sized based on the estimated demand. Yang et al. (2023) present a fleet sizing and charger infrastructure design for an autonomous fleet of electric taxis. Given a set of candidate locations, the problem consists of determining the number of chargers to be installed in each location, as well as sizing the fleet of electric vehicles operating. Solutions are evaluated through an extensive simulation accounting for link congestion and station capacity, and the problem is solved through Bayesian optimization.

Appendix C. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.ejor.2026.09.048.

Data availability

The input data, experiment configuration files, and detailed results for the ride-hailing experiments and the additional applications are publicly available at https://github.com/tms-publications/DSO-Nested-Clusters. The simulator used for the ride-hailing experiments has been made open source and is available as part of https://arxiv.org/abs/2609.30628OpenHail (Schettini et al., 2026).

References

Adenaw, L., & Lienkamp, M. (2020). A model for the data-based analysis and design of urban public charging infrastructure. In *2020 fifteenth international conference on ecological vehicles and renewable energies* (pp. 1–14). IEEE.

Agalianos, K., Ponis, S., Aretoulaki, E., Plakas, G., & Efthymiou, O. (2020). Discrete event simulation and digital twins: review and challenges for logistics. *Procedia Manufacturing*, *51*, 1636–1641.

Alrefaei, M. H., & Andradóttir, S. (1999). A simulated annealing algorithm with constant temperature for discrete stochastic optimization. *Management Science*, *45*(5), 748–764.

Anastasiadis, E., Angeloudis, P., Ainalis, D., Ye, Q., Hsu, P.-Y., Karamanis, R., Escribano Macias, J., & Stettler, M. (2021). On the selection of charging facility locations for EV-based ride-hailing services: A computational case study. *Sustainability*, *13*(1), 168.

Arslan, O., Karaşan, O. E., Mahjoub, A. R., & Yaman, H. (2019). A branch-and-cut algorithm for the alternative fuel refueling station location problem with routing. *Transportation Science*, *53*(4), 1107–1125.

Ball, R., Branke, J., & Meisel, S. (2018). Optimal sampling for simulated annealing under noise. *INFORMS Journal on Computing*, *30*(1), 200–215.

Ball, R., Branke, J., & Meisel, S. (2024). Re-use of samples in stochastic annealing. *Computers & Operations Research*, *164*, Article 106543. http://dx.doi.org/10.1016/j.cor.2024.106543.

Bi, R., Xiao, J., Pelzer, D., Ciechanowicz, D., Eckhoff, D., & Knoll, A. (2017). A simulation-based heuristic for city-scale electric vehicle charging station placement. In *2017 IEEE 20th international conference on intelligent transportation systems* (pp. 1–7). IEEE.

Boujelben, M. K., & Gicquel, C. (2019). Efficient solution approaches for locating electric vehicle fast charging stations under driving range uncertainty. *Computers & Operations Research*, *109*, 288–299.

Branke, J., Meisel, S., & Schmidt, C. (2008). Simulated annealing in the presence of noise. *Journal of Heuristics*, *14*(6), 627–654.

Can, B., Beham, A., & Heavey, C. (2008). A comparative study of genetic algorithm components in simulation-based optimisation. In *2008 winter simulation conference* (pp. 1829–1837). IEEE.

Capar, I., Kuby, M., Leon, V. J., & Tsai, Y.-J. (2013). An arc cover–path-cover formulation and strategic analysis of alternative-fuel station locations. *European Journal of Operational Research*, *227*(1), 142–151.

Chen, C.-H., & Lee, L. H. (2011). *Stochastic Simulation Optimization: An Optimal Computing Budget Allocation*. World Scientific.

Chen, C.-H., Lin, J., Yücesan, E., & Chick, S. E. (2000). Simulation budget allocation for further enhancing the efficiency of ordinal optimization. *Discrete Event Dynamic Systems: Theory and Applications*, *10*(3), 251–270.

Chick, S. E., & Inoue, K. (2001). New two-stage and sequential procedures for selecting the best simulated system. *Operations Research*, *49*(5), 732–743.

Cilio, L., & Babacan, O. (2021). Allocation optimisation of rapid charging stations in large urban areas to support fully electric taxi fleets. *Applied Energy*, *295*, Article 117072.

Costa, A., Alfieri, A., Matta, A., & Fichera, S. (2015). A parallel tabu search for solving the primal buffer allocation problem in serial production systems. *Computers & Operations Research*, *64*, 97–112.

de Sousa Junior, W. T., Montevechi, J. A. B., de Carvalho Miranda, R., & Campos, A. T. (2019). Discrete simulation-based optimization methods for industrial engineering problems: A systematic literature review. *Computers & Industrial Engineering*, *128*, 526–540.

DIMACS (2022). 12Th DIMACS implementation challenge: Vehicle routing problems. Online; (Accessed 07 July 2025). Full author: Center for Discrete Mathematics and Theoretical Computer Science (DIMACS), http://dimacs.rutgers.edu/programs/challenge/vrp/.

Eckman, D. J., & Feng, M. B. (2018). Green simulation optimization using likelihood ratio estimators. In *2018 winter simulation conference* (pp. 2049–2060). IEEE.

Eckman, D. J., & Henderson, S. G. (2018). Reusing search data in ranking and selection: What could possibly go wrong? *ACM Transactions on Modeling and Computer Simulation*, *28*(3), 18.

Fornell, C., & Cha, J. (1994). Partial least squares. In R. P. Bagozzi (Ed.), *Advanced Methods of Marketing Research* (pp. 52–78). Blackwell.

Gan, V. J., Wong, H., Tse, K. T., Cheng, J. C., Lo, I. M., & Chan, C. M. (2019). Simulation-based evolutionary optimization for energy-efficient layout plan design of high-rise residential buildings. *Journal of Cleaner Production*, *231*, 1375–1388.

Godbersen, G., Kolisch, R., & Schiffer, M. (2024). Robust charging network planning for metropolitan taxi fleets. *Transportation Science*, *58*(2), 295–314.

Göpfert, P., & Bock, S. (2019). A Branch&Cut approach to recharging and refueling infrastructure planning. *European Journal of Operational Research*, *279*(3), 808–823.

Gurobi Optimization, LLC (2023). Gurobi optimizer reference manual. URL https://www.gurobi.com.

He, X., & Niyogi, P. (2003). Locality preserving projections. In *Advances in Neural Information Processing Systems 16*.

Heinzl, B., & Kastner, W. (2020). A general variable neighborhood search for simulation-based energy-aware flow shop scheduling. In *2020 summer simulation conference* (pp. 1–12).

Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, *12*(1), 55–67.

Hong, L. J., & Nelson, B. L. (2006). Discrete optimization via simulation using COMPASS. *Operations Research*, *54*(1), 115–129.

Hosseini, M., Rahmani, A., & Hooshmand, F. (2022). A robust model for recharging station location problem. *Operational Research*, *22*, 4397–4440.

Javanroodi, K., Nik, V. M., & Mahdavinejad, M. (2019). A novel design-based optimization framework for enhancing the energy efficiency of high-rise office buildings in urban areas. *Sustainable Cities and Society*, *49*, Article 101597.

Jia, Y., Zhao, Y., Guo, Z., Xin, Y., & Chen, H. (2017). Optimizing electric taxi charging system: A data-driven approach from transport energy supply chain perspective. In *2017 IEEE electrical power and energy conference* (pp. 1–6). IEEE.

Kchaou-Boujelben, M. (2021). Charging station location problem: A comprehensive review on models and solution approaches. *Transportation Research Part C: Emerging Technologies*, *132*, Article 103376.

Kiatsupaibul, S., Smith, J. E., & Zabinsky, Z. B. (2018). Single observation adaptive search for continuous simulation optimization. *Operations Research*, *66*(6), 1713–1727.

Kim, J.-G., & Kuby, M. (2012). The deviation-flow refueling location model for optimizing a network of refueling stations. *International Journal of Hydrogen Energy*, *37*(6), 5406–5420.

Kong, W., Luo, Y., Feng, G., Li, K., & Peng, H. (2019). Optimal location planning method of fast charging station for electric vehicles considering operators, drivers, vehicles, traffic flow and power grid. *Energy*, *186*, Article 115826.

Kuby, M., & Lim, S. (2005). The flow-refueling location problem for alternative-fuel vehicles. *Socio-Economic Planning Sciences*, *39*(2), 125–145.

Kuby, M., & Lim, S. (2007). Location of alternative-fuel stations using the flow-refueling location model and dispersion of candidate sites on arcs. *Networks and Spatial Economics*, *7*, 129–152.

Kullman, N. (2022). Pyhailing. URL https://pypi.org/project/pyhailing/, Online; version 0.0.9; (Accessed 07 July 2025).

Kullman, N. D., Cousineau, M., Goodson, J. C., & Mendoza, J. E. (2022). Dynamic ride-hailing with electric vehicles. *Transportation Science*, *56*(3), 775–794.

Lee, C., & Han, J. (2017). Benders-and-price approach for electric vehicle charging station location problem under probabilistic travel range. *Transportation Research Part B: Methodological*, *106*, 130–152.

Márquez-Fernández, F. J., Bischoff, J., Domingues-Olavarría, G., & Alaküla, M. (2019). Using multi-agent transport simulations to assess the impact of ev charging infrastructure deployment. In *2019 IEEE transportation electrification conference and expo* (pp. 1–6). IEEE.

MirHassani, S., & Ebrazi, R. (2013). A flexible reformulation of the refueling station location problem. *Transportation Science*, *47*(4), 617–628.

Nelson, B. L. (2010). Optimization via simulation over discrete decision variables. In *Tutorials in operations research, 7* (pp. 193–207). Informs.

Nelson, B. L., Swann, J., Goldsman, D., & Song, W. (2001). Simple procedures for selecting the best simulated system when the number of alternatives is large. *Operations Research*, *49*(6), 950–963.

Pagani, M., Korosec, W., Chokani, N., & Abhari, R. S. (2019). User behaviour and electric vehicle charging infrastructure: An agent-based model assessment. *Applied Energy*, *254*, Article 113680.

Pan, A., Zhao, T., Yu, H., & Zhang, Y. (2019). Deploying public charging stations for electric taxis: A charging demand simulation embedded approach. *IEEE Access*, *7*, 17412–17424.

Pei, L., Nelson, B. L., & Hunter, S. (2018). A new framework for parallel ranking & selection using an adaptive standard. In *2018 winter simulation conference* (pp. 2201–2212). IEEE.

Rosen, S., Salemi, P., Wickham, B., Williams, A., Harvey, C., Catlett, E., Taghiyeh, S., & Xu, J. (2016). Parallel empirical stochastic branch and bound for large-scale discrete optimization via simulation. In *2016 winter simulation conference* (pp. 626–637). IEEE.

Schettini, T., Kullman, N. D., & Mendoza, J. E. (2026). OpenHail: An event-driven gymnasium environment for electric ride-hailing fleet control. http://dx.doi.org/10.48550/arXiv.2609.30628, arXiv:2609.30628, URL https://arxiv.org/abs/2609.30628.

Schneider, T. W. (2019). Taxi and ridehailing usage in New York City. URL https://toddwschneider.com/dashboards/nyc-taxi-ridehailing-uber-lyft-data/, Online; (Accessed 07 July 2025).

Shi, L., & Ólafsson, S. (2000a). Nested partitions method for global optimization. *Operations Research*, *48*(3), 390–407.

Shi, L., & Ólafsson, S. (2000b). Nested partitions method for stochastic optimization. *Methodology and Computing in Applied Probability*, *2*, 271–291.

Sweda, T., & Klabjan, D. (2011). An agent-based decision support system for electric vehicle charging infrastructure deployment. In *2011 IEEE vehicle power and propulsion conference* (pp. 1–5). IEEE.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B. Statistical Methodology*, *58*(1), 267–288.

Tu, W., Li, Q., Fang, Z., Shaw, S.-l., Zhou, B., & Chang, X. (2016). Optimizing the locations of electric taxi charging stations: A spatial–temporal demand coverage approach. *Transportation Research Part C: Emerging Technologies*, *65*, 172–189.

Uhlig, T., & Rose, O. (2011). Simulation-based optimization for groups of cluster tools in semiconductor manufacturing using simulated annealing. In *2011 winter simulation conference* (pp. 1852–1863). IEEE.

Viswanathan, V., Zehe, D., Ivanchev, J., Pelzer, D., Knoll, A., & Aydt, H. (2016). Simulation-assisted exploration of charging infrastructure requirements for electric vehicles in urban environments. *Journal of Computational Science*, *12*, 1–10.

Wang, Y.-W., & Lin, C.-C. (2009). Locating road-vehicle refueling stations. *Transportation Research Part E: Logistics and Transportation Review*, *45*(5), 821–829.

Wang, H., Pasupathy, R., & Schmeiser, B. W. (2013). Integer-ordered simulation optimization using R-SPLINE: Retrospective search with piecewise-linear interpolation and neighborhood enumeration. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, *23*(3), 1–24.

Wang, Y., Shi, J., Wang, R., Liu, Z., & Wang, L. (2018). Siting and sizing of fast charging stations in highway network with budget constraint. *Applied Energy*, *228*, 1255–1271.

Wang, H., Zhao, D., Cai, Y., Meng, Q., & Ong, G. P. (2021). Taxi trajectory data based fast-charging facility planning for urban electric taxi systems. *Applied Energy*, *286*, Article 116515.

Winkelhaus, S., & Grosse, E. H. (2020). Logistics 4.0: A systematic review towards a new logistics system. *International Journal of Production Research*, *58*(1), 18–43.

Wolbertus, R., van den Hoed, R., Kroesen, M., & Chorus, C. (2021). Charging infrastructure roll-out strategies for large scale introduction of electric vehicles in urban areas: An agent-based simulation study. *Transportation Research Part A: Policy and Practice*, *148*, 262–285.

Wu, F., & Sioshansi, R. (2017). A stochastic flow-capturing model to optimize the location of fast-charging stations with uncertain electric vehicle flows. *Transportation Research Part D: Transport and Environment*, *53*, 354–376.

Xi, X., Sioshansi, R., & Marano, V. (2013). Simulation–optimization model for location of a public electric vehicle charging infrastructure. *Transportation Research Part D: Transport and Environment*, *22*, 60–69.

Xu, W. L., & Nelson, B. L. (2013). Empirical stochastic branch-and-bound for optimization via simulation. *IIE Transactions*, *45*(7), 685–698.

Xu, J., Nelson, B. L., & Hong, L. J. (2013). An adaptive hyperbox algorithm for high-dimensional discrete optimization via simulation problems. *INFORMS Journal on Computing*, *25*(1), 133–146.

Yang, J., Levin, M. W., Hu, L., Li, H., & Jiang, Y. (2023). Fleet sizing and charging infrastructure design for electric autonomous mobility-on-demand systems with endogenous congestion and limited link space. *Transportation Research Part C: Emerging Technologies*, *152*, Article 104172.

Yıldız, B., Arslan, O., & Karaşan, O. E. (2016). A branch and price approach for routing and refueling station location model. *European Journal of Operational Research*, *248*(3), 815–826.

Yıldız, B., Olcaytu, E., & Şen, A. (2019). The urban recharging infrastructure design problem with stochastic demands and capacitated charging stations. *Transportation Research Part B: Methodological*, *119*, 22–44.

Zang, H., Fu, Y., Chen, M., Shen, H., Miao, L., Zhang, S., Wei, Z., & Sun, G. (2018). Bi-level planning model of charging stations considering the coupling relationship between charging stations and travel route. *Applied Sciences*, *8*(7), 1130.

Zhang, H., Sheppard, C. J., Lipman, T. E., Zeng, T., & Moura, S. J. (2020). Charging infrastructure demands of shared-use autonomous electric vehicles in urban areas. *Transportation Research Part D: Transport and Environment*, *78*, Article 102210.

Zhuge, C., & Shao, C. (2018). Agent-based modelling of locating public transport facilities for conventional and electric vehicles. *Networks and Spatial Economics*, *18*, 875–908.