

DETERMINISTIC NEIGHBORHOOD ROTATION (DNR) FOR CATEGORICAL VARIABLES IN DERIVATIVE-FREE OPTIMIZATION*

J. F. A. MADEIRA[†]

Abstract. Categorical variables arise in many optimization problems but lack the order and metric structure required by geometric derivative-free optimization (DFO) methods. Existing approaches either impose artificial orderings, which bias the search, or rely on stochastic exploration, compromising reproducibility.

We propose a finite deterministic framework based on *deterministic neighborhood rotation* (DNR), where categorical values are represented canonically while neighborhood structures vary through a deterministic permutation schedule. A deterministic low-discrepancy sequence with dense image is used to ensure pairwise rank-1 adjacency coverage at the schedule level, with the Sobol-rank construction as a practical instance. A canonical cache provides coordinate-wise local optimality certification when all one-variable alternatives of the incumbent have been evaluated; after convergence, this verification requires no additional function evaluations.

The resulting method is fully deterministic, reproducible, and cache-efficient. Numerical results on standard benchmarks demonstrate robustness with respect to problem size, and confirm that the performance gap of integer-encoding search is primarily structural rather than due to initialization.

Key words. derivative-free optimization, categorical variables, mixed-variable optimization, deterministic methods, permutation-based neighborhoods

AMS subject classifications. 90C56, 90C30

1. Introduction. Many real-world optimization problems involve decision variables of mixed nature, combining continuous, integer, discrete numerical, and categorical components. Such problems arise naturally in engineering design, materials selection, system configuration, and algorithm tuning, among other application domains [2, 5]. While continuous and discrete numerical variables admit natural geometric structures, categorical variables—also referred to in the literature as qualitative or nominal variables—represent qualitatively different choices that lack intrinsic order or metric structure [2, 21, 14].

Geometric derivative-free optimization (DFO) methods are particularly attractive for black-box and simulation-based problems, where derivatives are unavailable or unreliable [17, 6]. These methods rely on geometric notions such as neighborhoods, local refinement, and step-size control, and have been shown to be effective for continuous and, to some extent, integer and discrete numerical variables [18, 4]. However, the integration of categorical variables into geometric DFO frameworks remains challenging.

Existing approaches for handling categorical variables typically follow one of two paths. On the one hand, categorical values are often encoded as integers and treated as ordered variables, implicitly imposing an artificial geometry that may bias the search [18, 10]. On the other hand, many heuristic and metaheuristic methods rely on random neighbor selection to explore categorical alternatives, sacrificing determinism and reproducibility [9, 16]. Alternative strategies based on latent embeddings or one-

*

Funding: The author acknowledges the support of Fundação para a Ciência e a Tecnologia (FCT), Portugal, through LAETA (project UID/50022/2025).

[†]IDMEC-IST, Instituto Superior Técnico, Universidade de Lisboa, Av. Rovisco Pais, 1040-001 Lisboa, Portugal, and ISEL-IPL, Rua Conselheiro Emídio Navarro, 1, 1959-007 Lisboa, Portugal (aguilarmadeira@tecnico.ulisboa.pt).

hot encodings have been proposed in surrogate-based optimization [8, 22], but these are less compatible with geometric direct-search frameworks. Recent extensions of the MADS framework [1] and CMA-ES [12] address categorical variables in mixed-variable settings by constructing problem-specific geometries or employing stochastic operators; these are discussed in Section 3.3. A common feature of these approaches is that they often conflate two distinct concerns: the *representation* of categorical values and the *geometric structure* used to explore them. When encoding strategies developed for statistical models are adopted in optimization, they may inadvertently impose distance and neighborhood structures that have no justification in the original problem. As a result, there is currently no widely accepted treatment of categorical variables that is simultaneously deterministic, fully reproducible, and free of imposed artificial geometry.

Determinism and reproducibility are not merely theoretical concerns. In many engineering and industrial contexts, optimization results must be auditable, repeatable, and independent of random seeds or platform-specific pseudo-random number generators. This requirement is at odds with stochastic approaches, yet fixed artificial orderings introduce systematic biases that are equally undesirable. Bridging this gap remains an open problem. The present work focuses on the fully feasible categorical product space; extensions to constrained categorical domains, where certain combinations of categories are forbidden, are left for future work.

In this work, we propose a simple and fully deterministic framework for handling categorical variables in direct-search optimization. The proposed approach is based on *permutation-induced neighborhood rotation*. Categorical values are represented canonically and remain fixed throughout the optimization process, while their neighborhood structure is defined by a permutation that changes periodically according to a deterministic schedule. This mechanism enables structured, deterministic exposure of all categorical alternatives without imposing a fixed artificial order or relying on random number generators. At its core, the proposed framework separates categorical identity from categorical exploration and replaces both artificial ordering and randomness by a finite deterministic framework with a rotating neighborhood structure and a coordinate-wise certification guarantee. In contrast to approaches that embed categorical variables into an artificial geometry — whether through integer encoding, learned distances, or graph structures — the present work avoids imposing any persistent neighborhood structure: deterministic structural diversity is achieved through controlled variation of the permutation-induced adjacency relation rather than structural imposition.

The main contributions of this paper are:

- a principled framework for categorical variables based on six design principles that ensure canonical representation, separation between values and neighborhoods, and controlled exploration without randomness;
- two deterministic rotation schedules — a cyclic-shift schedule and a Sobol-rank schedule — with a schedule-level pairwise adjacency coverage result (Proposition 4.8) characterising the expressive power of dense permutation schedules; this result is logically independent of algorithmic correctness;
- a complete DNR-Search algorithm in which permutation changes are triggered by algorithmic state ($r = 1$ and poll failure) rather than a fixed epoch counter, with a canonical cache, a formal coordinate-wise local optimality certificate (Proposition 4.16), and a dual stopping criterion;
- practical implementation guidelines and numerical validation on standard benchmark problems with all-categorical variable representations.

The remainder of this paper is organized as follows. Section 2 reviews the main types of decision variables in mixed-variable optimization and highlights the structural challenges posed by categorical variables. Section 3 discusses classical approaches to handling categorical variables and their limitations in deterministic geometric settings. Section 4 presents the DNR mechanism together with the conceptual requirements underlying its design, including the permutation-based neighborhood construction, its deterministic rotation schedules, and the coverage and optimality guarantees. The distinction between deterministic neighborhood rotation and stochastic exploration, together with practical implementation guidelines, is discussed in Section 5. Section 6 presents numerical experiments validating the algorithm on standard benchmarks. Section 7 concludes the paper with a summary and directions for future work.

2. Variable Types in Mixed-Variable Optimization. Mixed-variable optimization problems involve decision variables of heterogeneous types. Following the taxonomy of [2, 3], we distinguish four fundamental types based on their mathematical structure; detailed descriptions of the first three are provided in Appendix B and here we focus on the structural distinction of categorical variables. We consider the general problem

$$(2.1) \quad \min_{x \in \Omega} f(x), \quad \Omega = \Omega_C \times \Omega_I \times \Omega_D \times \Omega_K,$$

where Ω_C , Ω_I , Ω_D , and Ω_K denote the domains of continuous, integer, discrete numerical, and categorical variables, respectively. Continuous, integer, and discrete numerical variables all admit a natural order and metric structure (and the first two a normalization to $[0, 1]^n$) that support geometric search [5, 18, 2]. Categorical variables are structurally different.

2.1. Categorical Variables. A categorical variable x_i takes values in a finite unordered set

$$(2.2) \quad x_i \in \mathcal{C}_i = \{c_1, c_2, \dots, c_{m_i}\},$$

where the elements c_k represent qualitative labels rather than numerical quantities. Typical examples include material choices, algorithmic configurations, or model selections.

Categorical variables are fundamentally different from the previous types:

- they possess no natural order;
- they admit no intrinsic metric structure;
- they are not refinable in the usual geometric sense.

Only equality and inequality relations are meaningful for categorical variables. Any numerical encoding that imposes an order or a notion of distance is therefore artificial and does not reflect the underlying problem semantics [21]. As noted by Audet and Hare [2], *categorical variables require special treatment because standard notions of distance and direction do not apply.*

For algorithmic purposes, categories are often represented by indices $k \in \{1, \dots, m_i\}$. The associated normalized encoding

$$(2.3) \quad y_i = \frac{k-1}{m_i-1} \in \left\{0, \frac{1}{m_i-1}, \dots, 1\right\}$$

serves purely as a positional identifier. It must not be interpreted as a coordinate in a metric space. This distinction is central to the categorical treatment proposed in this work.

2.2. Summary and Implications. Table 2.1 summarizes the structural properties of the variable types considered.

TABLE 2.1
Structural properties of variable types in mixed-variable optimization.

Type	Domain	Order	Metric	Refinable
Continuous (C)	$[l, u] \subset \mathbb{R}$	Yes	Yes	Yes
Integer (I)	$\{l, \dots, u\} \subset \mathbb{Z}$	Yes	Yes (step 1)	Limited
Discrete (D)	$\{v_1, \dots, v_m\}$ ordered	Yes	Yes (step Δ)	Limited
Categorical (K)	$\{c_1, \dots, c_m\}$ unordered	No	No	No

The key observation is that categorical variables lack the mathematical structure upon which geometric optimization algorithms rely. While continuous, integer, and discrete numerical variables can be treated within a unified geometric framework, categorical variables require fundamentally different considerations.

This structural gap motivates the need for a principled categorical treatment that does not impose artificial order or metric, enables deterministic and finite exposure of all categories, and preserves reproducibility. The next section reviews classical approaches to categorical variables and highlights their limitations in deterministic geometric settings.

3. Classical Treatment of Categorical Variables. Several approaches have been proposed for handling categorical variables in optimization. This section reviews the most common strategies and highlights their limitations, with emphasis on settings where *determinism* and *geometric consistency* are required.

3.1. Integer Encoding. The simplest approach assigns consecutive integers to categories,

$$(3.1) \quad c_k \mapsto k, \quad k \in \{1, 2, \dots, m\},$$

so that categorical variables can be handled by mixed-integer frameworks [18].

While straightforward, integer encoding imposes an *artificial order*: category c_2 becomes “between” c_1 and c_3 , suggesting that $c_1 \rightarrow c_2$ is a smaller change than $c_1 \rightarrow c_3$. This is semantically unjustified whenever categories have no intrinsic ordering. For geometric algorithms, this can lead to:

- **label dependence:** the search trajectory depends on the arbitrary labeling chosen by the user;
- **biased neighborhood exploration:** categories adjacent in the imposed order are favored over others;
- **fictitious locality:** the algorithm may exploit a local structure that does not exist in the original problem.

Some works attempt to reduce labeling bias by trying multiple labelings (e.g., randomized relabeling) [10], but this typically sacrifices full determinism. The empirical consequences of label dependence under a non-monotone categorical embedding are demonstrated in Section 6, where integer-encoding search (IE) is used as a deterministic baseline.

3.2. Random Neighbor Selection. Metaheuristics such as genetic algorithms [9, 7] and simulated annealing [16] typically handle categorical variables through stochastic operators, e.g., mutation (randomly switching categories) and crossover (in-

heriting a category from a randomly selected parent). This avoids imposing artificial structure and, in principle, treats all categories symmetrically.

However, random neighbor selection is fundamentally *stochastic*:

- **lack of robust reproducibility:** different runs produce different trajectories and outcomes;
- **implementation dependence:** results may depend on pseudo-random generators, seeding, and platform-specific details;
- **limited compatibility with geometric DFO:** stochastic switching does not naturally integrate with deterministic neighborhood refinement.

This tension between avoiding artificial order and maintaining determinism is a key motivation for the present work. The statistical variability and lack of certification inherent to random neighbor selection are quantified in Section 6, where a memory-free stochastic baseline (RNS) is included as a lower-bound comparator.

Additional encoding strategies that avoid an explicit integer order, including one-hot representations and latent variable embeddings, are discussed in Appendix C; their limitations in geometric DFO settings are noted there.

3.3. Graph-based and distance-based direct-search approaches. An alternative deterministic treatment of categorical variables within geometric DFO has been proposed through graph-based and distance-based extensions of MADS, notably CatMADS [1]. In this approach, categorical variables are endowed with a problem-specific distance constructed by cross-validation on an initial design of experiments. This distance induces a neighborhood structure: categories closest under the learned metric are polled first. CatMADS preserves determinism and provides a comprehensive convergence analysis, introducing four progressively stronger notions of mixed-variable local optimality. A prototype implementation is benchmarked against state-of-the-art solvers on 32 mixed-variable test problems and demonstrates competitive empirical performance.

While this strategy is well-motivated and principled, it operates under fundamentally different structural assumptions from the present work. CatMADS embeds categorical variables into a metric space by constructing a problem-specific distance, and then applies mesh-based polling within that space. The method therefore assumes — or constructs — a geometry on the categorical domain. This introduces two concerns in settings where categories are purely nominal and no intrinsic structure exists:

- the imposed distance may bias the search if it does not accurately reflect the true problem semantics — a risk amplified when the initial design of experiments is small or the objective landscape is irregular;
- the neighborhood structure, once constructed, remains fixed (or slowly adaptive), which may prevent systematic exposure of all categorical alternatives if the learned metric assigns large distances to globally important transitions.

A related stochastic approach, CatCMA with Margin [12], extends CatCMA [13] to handle continuous, integer, and categorical variables jointly via a joint Gaussian-categorical distribution with modified margin correction. This method is empirically strong but inherently stochastic: reproducibility depends on random seeds and platform-specific generators.

The present work takes a structurally different stance. Rather than constructing a geometry — fixed or learned — for categorical variables, DNR-Search works entirely in the *nominal categorical space*: it never embeds categories into a metric space, never computes distances between categories, and never imposes a persistent neighborhood

structure. Neighborhoods are induced by permutations that rotate deterministically over time, ensuring that all categorical pairs can eventually be exposed without any permanent adjacency bias and without requiring an initial evaluation budget for structure construction.

This distinction means that DNR-Search and CatMADS are not directly comparable at the algorithmic or theoretical level, for two reasons. First, CatMADS is designed for mixed continuous-categorical spaces and leverages a geometric structure on the continuous variables; DNR-Search operates exclusively on finite categorical product spaces without any imposed metric. Second, restricting CatMADS to purely categorical instances would require fixing all continuous variables — a non-standard use that does not reflect the method’s intended operating regime, and would not produce a meaningful like-for-like comparison. CatMADS requires a distance structure and targets the full mixed continuous-categorical problem with continuous convergence theory; DNR-Search targets the purely nominal categorical setting and provides finite-domain correctness guarantees without metric assumptions. A fair empirical comparison therefore requires a mixed-variable extension of DNR-Search, identified as a priority direction for future work (Section 7). The two paradigms are complementary rather than competing: the DNR mechanism could serve as a deterministic, bias-free categorical component within CatMADS or similar frameworks, replacing their distance-based categorical polling.

3.4. Summary of Limitations. Table 3.1 summarizes the classical approaches and their main limitations in deterministic geometric settings. The fundamental trade-off is clear: methods that avoid imposing a fixed artificial order sacrifice determinism or geometric compatibility, while the fully deterministic geometric option (integer encoding) introduces label dependence and biased locality. Figure 3.1 illustrates the three paradigms geometrically.

TABLE 3.1
Classical approaches to categorical variables and their limitations.

Approach	Imposes order	Deterministic	Geometric
Integer encoding	Yes	Yes	Yes (artificial geometry)
One-hot encoding	No	Yes	No
Random selection	No	No	No
Latent variables	No	No (typically)	Partially (model-induced)

The key observation is that classical encodings conflate two distinct concerns: categorical values require a *canonical representation* for identity and caching, while categorical exploration requires a *neighborhood structure* for generating alternatives. This motivates the central question of this paper:

Can categorical variables be explored systematically and deterministically, without imposing a fixed artificial order, while remaining compatible with geometric derivative-free optimization?

Our answer is *yes*, via *deterministic neighborhood rotation* — neighborhoods defined through deterministic permutations that vary over time, providing full reproducibility and natural integration with geometric direct-search mechanisms.

4. Deterministic Permutation-Based Neighborhoods. The central principle of this work is the *separation of categorical identity from categorical exploration*. Categorical values are represented by fixed canonical identifiers used for point identity,

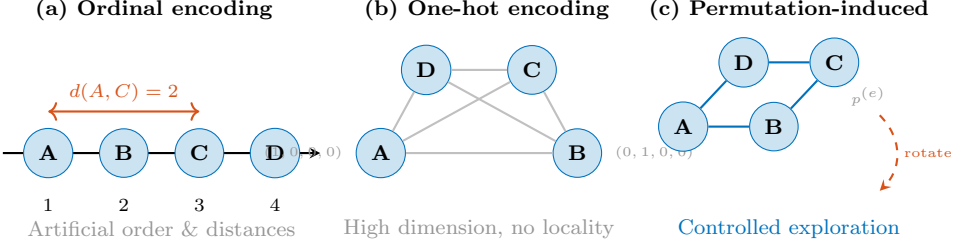


FIG. 3.1. Comparison of categorical variable treatments. (a) Ordinal encoding imposes artificial order and distances: $d(A, C) = 2$ has no semantic meaning. (b) One-hot encoding embeds categories in a high-dimensional simplex where all pairs are equidistant, destroying locality. (c) The proposed approach (introduced in Section 4) defines a minimal cyclic neighborhood structure whose ordering changes deterministically over time.

comparison, and caching, while the neighborhood structure used to explore alternatives varies deterministically over time. This separation eliminates the need to impose any persistent artificial order or metric on categorical variables, while preserving compatibility with geometric DFO frameworks.

We realize this construction through *deterministic permutation-based neighborhoods*. At each stage, a permutation induces a minimal cyclic neighborhood and is updated by a state-triggered deterministic rule. This yields two logically independent components: an *exploration component*, governed by the permutation schedule, and a *certification component*, based on the canonical cache and the stopping criterion. The former yields a coverage property (Proposition 4.8); the latter yields a coordinate-wise local optimality certificate (Proposition 4.16). Algorithmic correctness depends only on the latter. The correctness of the method is independent of the specific choice of permutation schedule.

The separation principle leads to the following design requirements, which are referred to throughout the paper. **P1** (canonical representation): each categorical variable is represented by a fixed canonical identifier, invariant throughout the search and used consistently for point identity, caching, and comparison. **P2** (separation): the value being evaluated does not change; only its neighborhood changes — preventing the conflation of representation and exploration inherent to classical encodings. **P3** (no intrinsic order or metric): any neighborhood relation is auxiliary and algorithm-defined, not a measure of similarity between categories. **P4** (determinism): all components are fully deterministic; no stochastic operators are introduced. **P5** (controlled exploration): exploration is achieved through controlled variation of the neighborhood structure, independent of the arbitrary labeling assigned to categories. **P6** (DFO compatibility): the construction integrates with normalized search spaces, cache-based evaluation, and step-size mechanisms.

The following standing assumptions hold throughout this section and the rest of the paper. Their logical relationship to the propositions and algorithm is summarised in Figure 4.1.

ASSUMPTION 4.1 (Standing assumptions).

- (i) Finite cardinality: each categorical variable x_i has finite domain $C_i = \{c_1, \dots, c_{m_i}\}$ with $m_i \geq 2$.
- (ii) Homogeneous cardinality: the theoretical development in this section is stated for the case of a common number of categories $m_i = m$ for all variables. Extension to heterogeneous cardinalities m_i requires independent permutation

constructions per variable and is not pursued here; the relevant adaptations are $r_i(\alpha) = \max(1, \text{round}(\alpha m_i))$ and $N_{\text{stop}} = \sum_i (m_i - 1)$.

- (iii) Deterministic evaluation: the objective function f is deterministic; every evaluation of the same point returns the same value.
- (iv) Full domain feasibility: the objective function f is defined for every point $h \in \{1, \dots, m\}^n$; no categorical combination is infeasible.

Assumption 4.1 separates structural and algorithmic requirements, which are used differently in the two components of the analysis.

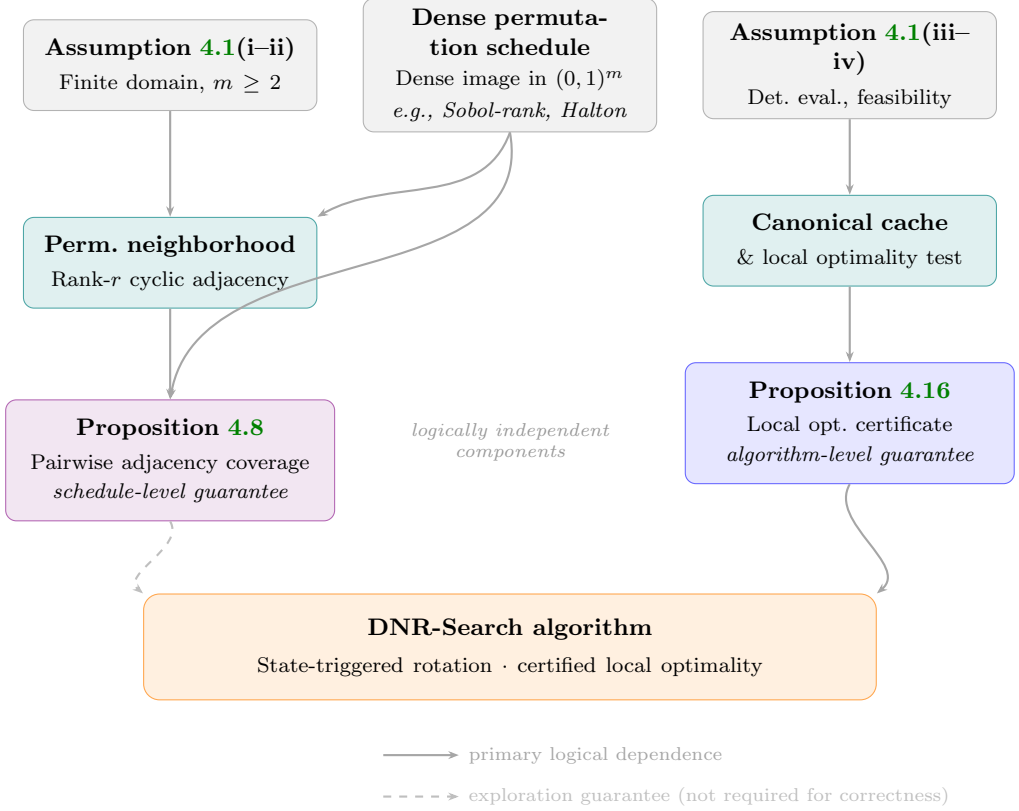


FIG. 4.1. Logical dependency structure of the DNR-Search theoretical framework. Solid arrows denote primary logical dependence, while the dashed arrow represents an exploration guarantee that is not required for correctness. Proposition 4.8 establishes a schedule-level pairwise adjacency coverage property for any dense permutation schedule, with the Sobol-rank construction as the recommended instance. Proposition 4.16 provides an algorithm-level certificate of coordinate-wise categorical local optimality, based solely on the canonical cache and the stopping criterion. The two propositions are logically independent and address distinct roles: exploration versus certification. The DNR-Search algorithm combines both components through a state-triggered permutation rotation and a certified stopping mechanism.

4.1. Canonical encoding and permutation-induced neighborhood. Let x_i be a categorical variable taking values in a finite set

$$\mathcal{C}_i = \{c_1, \dots, c_{m_i}\}.$$

For algorithmic purposes, each category is represented by an index $k \in \{1, \dots, m_i\}$. The *reconstruction map* $\psi : \{1, \dots, m\}^n \rightarrow \Omega_K$ maps canonical index vectors back to categorical assignments in the original space; it is assumed well-defined, injective, and fixed throughout the optimization, so that distinct index vectors represent distinct categorical decisions. This index is a label only and does not imply any intrinsic order.

DEFINITION 4.2 (Canonical encoding). *For a categorical variable with $m \geq 2$ categories represented by indices $k \in \{1, \dots, m\}$ (under Assumption 4.1(ii) we write $m_i = m$ for all variables), the canonical normalized encoding is the map $\eta : \{1, \dots, m\} \rightarrow [0, 1]$ defined by*

$$(4.1) \quad \eta(k) = \frac{k-1}{m-1}.$$

The value $\eta(k)$ is used solely as a *canonical identifier* in the normalized space, ensuring compatibility with scale-invariant algorithms that operate on $[0, 1]^n$. Importantly, the encoding in (4.1) is invariant throughout the optimization process and does not depend on any neighborhood structure. Consequently, point identity and caching for categorical variables are always performed using the canonical index k (or equivalently $\eta(k)$), rather than any position-dependent representation.

We now introduce an algorithmic neighborhood relation for categorical variables based on permutations.

DEFINITION 4.3 (Permutation-induced neighborhood of rank r). *Let $p = (p_1, \dots, p_m)$ be a permutation of $\{1, \dots, m\}$, interpreted cyclically (with $p_{m+1} = p_1$ and $p_0 = p_m$, indices taken modulo m). For a category $k \in \{1, \dots, m\}$, let j denote its position in p , i.e., $p_j = k$. For any integer $r \in \{1, \dots, \lfloor m/2 \rfloor\}$, the rank- r neighbors of k under p are*

$$(4.2) \quad h^+(k, p, r) = p_{(j+r-1 \bmod m)+1}, \quad h^-(k, p, r) = p_{(j-r-1 \bmod m)+1}.$$

The rank- r permutation-induced neighborhood of k is

$$(4.3) \quad \mathcal{N}_p^r(k) = \{h^-(k, p, r), h^+(k, p, r)\}.$$

The rank-1 neighborhood $\mathcal{N}_p(k) := \mathcal{N}_p^1(k) = \{p_{j-1}, p_{j+1}\}$ is the canonical case used in the coverage guarantee (Proposition 4.8).

Definition 4.3 induces a 2-regular cyclic graph on the set of categories at each rank r , with the rank-1 case being the finest resolution. The rank- r neighborhood is used by Algorithm 4.1 to explore categories at multiple scales: large r gives coarser, longer-range steps; $r = 1$ gives the finest, most local steps. As with rank-1, this neighborhood structure is *auxiliary* and algorithm-defined (Principle P3): it does not assert intrinsic similarity between categories at any rank.

Remark 4.4 (Independence from canonical identity). The permutation p affects only the neighborhood relation $\mathcal{N}_p(\cdot)$. It does not affect the canonical encoding $\eta(\cdot)$ used for point identity. Therefore, caching and point comparison remain consistent even if the permutation changes over time: a categorical value k always corresponds to the same canonical representation (4.1).

At this stage, the permutation p is assumed fixed. The following subsection introduces a deterministic mechanism to rotate the permutation over time, enabling structured exposure of categorical alternatives without imposing a fixed artificial order.

4.2. Deterministic permutation rotation. So far, the permutation p has been treated as fixed. However, fixing a single permutation throughout the optimization process would reintroduce label dependence: the neighborhood structure would permanently favor certain categorical transitions over others.

To address this issue, we introduce *deterministic permutation rotation*. Rather than changing the permutation on a fixed schedule, we use a *state-triggered* rule that ties the permutation change to the algorithmic state.

State-triggered permutation change. The permutation advances when the algorithm has exhausted the current permutation at maximum resolution. Specifically, a permutation is considered *exhausted* when the rank reaches its minimum value ($r = 1$) and the poll fails to improve the current point. At this moment, the step size is reset to its initial value α_0 and the permutation index is incremented.

Since $r(\alpha) = 1$ for all $\alpha < 1.5/m$, and α contracts geometrically under repeated poll failures, rank-1 is always eventually reached. An unsuccessful poll at $r = 1$ confirms that no rank-1 neighbour under the current permutation improves the incumbent; the permutation then advances.

The state-triggered mechanism eliminates the epoch size E as a free parameter: the permutation changes exactly when it should, after the algorithm has fully exploited the current neighborhood structure at the finest scale.

Two deterministic rotation schedules. To guarantee full determinism, permutations are generated without recourse to pseudo-random number generators. The simplest approach is a cyclic shift of the identity permutation. However, cyclic shifts preserve the *relative order* of all elements: a pair of categories that are not rank-1 adjacent in the initial permutation remain non-adjacent under all subsequent shifts. For $m \geq 4$, this means full pairwise rank-1 coverage cannot be achieved.

Remark 4.5 (Why cyclic shifts fail for $m \geq 4$). Under a cyclic shift by one position, element p_j moves to position p_{j-1} but the *relative order* of all elements is preserved: if p_j and p_k are not adjacent (i.e., $|j - k| \not\equiv 1 \pmod{m}$) in the initial permutation, they remain non-adjacent in every subsequent shift. For $m = 3$ every pair is at distance at most 1 cyclically, so full coverage is trivially achieved. For $m \geq 4$, non-adjacent pairs (e.g., positions 1 and 3 in $(1, 2, 3, 4)$) are never rank-1 adjacent under any cyclic shift, confirming that the cyclic schedule cannot provide the pairwise adjacency coverage guarantee of Proposition 4.8.

The recommended schedule, described next, avoids this limitation.

DEFINITION 4.6 (Sobol-rank permutation). *Let m be the number of categories and let $s \geq 0$ be a non-negative integer called the permutation index. The Sobol-rank permutation $p^{(s)}$ is constructed as follows:*

1. *Generate the s -th point $u = (u_1, \dots, u_m) \in (0, 1)^m$ of the m -dimensional Sobol sequence.*
2. *Define $p^{(s)}$ as the permutation that sorts the components of u in increasing order, breaking ties deterministically by increasing index.*

LEMMA 4.7 (Openness of the adjacency set). *For any two distinct categories $c, c' \in \{1, \dots, m\}$, the set*

$$A_{c,c'} = \{u \in (0, 1)^m : u_c \text{ and } u_{c'} \text{ are adjacent in the sorted order of } u\}$$

is nonempty and open in $(0, 1)^m$.

Proof. Nonemptiness: choose $u_c = 1/3$, $u_{c'} = 2/3$, and place all remaining $m - 2$ coordinates in $(0, 1/3) \cup (2/3, 1)$; then no coordinate falls strictly between u_c and $u_{c'}$,

so c and c' are adjacent in the sorted order. *Openness*: adjacency of c and c' is the condition that no coordinate $u_{c''}$, $c'' \notin \{c, c'\}$, satisfies $\min(u_c, u_{c'}) < u_{c''} < \max(u_c, u_{c'})$. This is a finite conjunction of strict inequalities, hence an open condition. \square

The following proposition concerns the *exploration* component of DNR-Search only and is logically independent of the stopping certificate (Proposition 4.16). Its conclusion does not affect the correctness of the stopping criterion, which is established independently through cache-based certification. Proposition 4.8 is *not* invoked in the proof of correctness or finite termination of DNR-Search; it characterises the expressive power of the schedule, not the correctness of the method. In particular, the algorithm may terminate without ever visiting the permutation s^* whose existence is guaranteed by the proposition. The low-discrepancy schedule provides structured variability across permutations without introducing any randomness. Under deterministic evaluation and the fixed algorithmic rules stated above, the sequence of evaluated points is uniquely determined by the initial point and the parameters. Specifically: variables are polled in the fixed order $i = 1, \dots, n$; for each variable the two trial points h_i^+, h_i^- are tested in that order; only strict improvement ($f(\psi(h')) < f^*$) is accepted; and ties in Sobol coordinate sorting are broken by increasing index.

PROPOSITION 4.8 (Pairwise adjacency coverage under a dense schedule). *Let $(u^{(s)})_{s \geq 0}$ be a sequence with dense image in $(0, 1)^m$, and let $p^{(s)}$ be the permutation induced by sorting the components of $u^{(s)}$ in increasing order, with ties broken deterministically by increasing index. Then, for any two distinct categories $c, c' \in \{1, \dots, m\}$, there exists a finite index $s^* < \infty$ such that c' is a rank-1 neighbor of c under $p^{(s^*)}$.*

Proof. By Lemma 4.7, $A_{c,c'}$ is nonempty and open. Since $(u^{(s)})_{s \geq 0}$ has dense image in $(0, 1)^m$ by hypothesis, there exists a finite s^* such that $u^{(s^*)} \in A_{c,c'}$. Hence c' is a rank-1 neighbor of c under $p^{(s^*)}$. \square

Remark 4.9 (Extension to rank- r coverage). The argument of Proposition 4.8 extends directly to rank- r adjacency for any fixed $r \in \{1, \dots, \lfloor m/2 \rfloor\}$, but this extension is not required in the present work.

Remark 4.10 (Dense schedules: Sobol and Halton). The Sobol sequence is a (t, m) -sequence in base 2; its star discrepancy satisfies $D_N^*(S) \rightarrow 0$ as $N \rightarrow \infty$ [20, Theorem 4.17], which implies uniform distribution and hence dense image in $(0, 1)^m$. The Halton sequence [11] satisfies the same density property. Both are therefore valid instances of the dense schedule required by Proposition 4.8; the Sobol-rank construction (Definition 4.6) is the recommended default owing to its superior uniformity properties in moderate dimensions [15].

Remark 4.11 (Schedule coverage versus algorithmic stopping). The pairwise adjacency coverage guarantee of Proposition 4.8 is a property of the permutation *schedule*: it states that every pair of categories will eventually become rank-1 adjacent under some permutation in the sequence. The DNR-Search algorithm may terminate much earlier, once coordinate-wise local optimality is certified via the canonical cache (Section 4.2). The theoretical guarantee provides an asymptotic exploration property; the stopping criterion provides a finite-time optimality certificate that does not require the schedule's pairwise adjacency coverage to be reached. Note further that the primary stopping criterion is strictly stronger than verifying only rank-1 neighbors under the current permutation: it requires that *all* $m - 1$ alternatives of each variable are present in the cache, regardless of the rank or permutation through which each was first evaluated.

Remark 4.12 (Geometric foundation and tight coverage bound). A formal geometric foundation for the transient neighborhood structure introduced here is developed in the companion work [?]. That work formalises mixed search spaces as weighted Cartesian products of metric spaces, introduces a hierarchy of local optimality (instantaneous, robust, and categorical), and proves a tight coverage bound: $\lceil (k-1)/2 \rceil$ cyclic orderings are both sufficient and minimal for full categorical adjacency coverage of a k -level variable, with an explicit constructive procedure based on a classical Hamiltonian decomposition. It also establishes uniform metric equivalence across all orderings, confirming that ordering rotation preserves the topology of the search space. The present algorithm can be viewed as a concrete DFO instantiation of that geometric framework.

We now present DNR-Search, a complete single-objective coordinate search algorithm that incorporates the DNR mechanism, a canonical cache, and dual stopping criteria. The algorithm is self-contained and does not require an external DFO solver. From this point onward, we restrict attention to the purely categorical case: all n decision variables are categorical ($q = n$).

Canonical cache.. The cache stores all evaluated points indexed by their canonical identity h , independently of the active permutation. This is the *only* data structure needed for both evaluation reuse and coverage verification (Principle P2). Throughout this section, we write $f(h)$ as shorthand for $f(\psi(h))$ whenever $h \in \{1, \dots, m\}^n$ is a canonical index vector; the map $\psi : \{1, \dots, m\}^n \rightarrow \Omega_K$ is the reconstruction map from canonical indices to categorical assignments, and is assumed well-defined and injective so that distinct index vectors represent distinct categorical decisions.

DEFINITION 4.13 (Canonical cache). *The canonical cache \mathcal{C} is an accumulative set of pairs $(h, f(\psi(h)))$, where $h \in \{1, \dots, m\}^n$ is the canonical index vector and $\psi(h)$ is the corresponding point in the original space. The cache is never reset, even when the incumbent h^* changes. A point h' is cache-matched if $(h', \cdot) \in \mathcal{C}$. When a trial point h' is proposed:*

- if h' is cache-matched, its stored value is returned without a new function evaluation;
- otherwise, $f(\psi(h'))$ is computed, the evaluation count is incremented, and $(h', f(\psi(h')))$ is added to \mathcal{C} .

For the coverage stopping criterion, a neighbour of the current h^ is considered covered if and only if it is present in the cache — regardless of when or through which permutation it was first evaluated.*

Rank function.. The step size $\alpha \in (0, 1]$ induces an integer rank that governs the neighborhood size:

$$(4.4) \quad r(\alpha) = \max\left(1, \min\left(\lfloor m/2 \rfloor, \text{round}(\alpha \cdot m)\right)\right).$$

The upper clip at $\lfloor m/2 \rfloor$ ensures that $r(\alpha)$ remains within the admissible range of Definition 4.3 for all $\alpha \in (0, 1]$. As $\alpha \rightarrow 0$, $r(\alpha) = 1$ for all $\alpha < 1.5/m$.

Remark 4.14 (Degeneracy at $m = 2$ and even m at maximum rank). When $m = 2$, the only admissible rank is $r = 1$, and the two categories are each other's unique neighbor under any permutation; the algorithm reduces to testing the single alternative of each variable. When m is even and $r = \lfloor m/2 \rfloor = m/2$, the forward and backward rank- r neighbors coincide for every category (since $j + m/2 \equiv j - m/2 \pmod{m}$), so $|\mathcal{N}_p^r(k)| = 1$. The algorithm remains correct in both cases; the set interpretation of $\mathcal{N}_p^r(k)$ handles the singleton automatically.

Stopping criteria. Two complementary criteria are used.

Primary criterion (full coordinate-neighbour coverage): the algorithm stops when every alternative $k' \neq h_i^*$ of each variable i at the current best point h^* has been evaluated and is present in the cache. This means that all $n \cdot (m - 1)$ single-variable perturbations of h^* have been tested at least once without improving f . Note that this criterion checks *all* alternatives of each variable, regardless of the rank or permutation through which each alternative was first reached — it is therefore strictly stronger than verifying only rank-1 neighbours under the current permutation.

We first formalise the notion of local optimality used in the stopping certificate.

DEFINITION 4.15 (Coordinate-wise categorical local optimum). *A point $h^* \in \{1, \dots, m\}^n$ is a coordinate-wise categorical local optimum of f if $f(h^*) \leq f(h)$ for every h obtained from h^* by replacing exactly one component h_i^* by some $k' \neq h_i^*$.*

PROPOSITION 4.16 (Cache-based local optimality certification). *Let $h^* \in \{1, \dots, m\}^n$ be the incumbent at termination. If every one-variable alternative of h^* is in the canonical cache with value $f \geq f(h^*)$, then h^* is a coordinate-wise categorical local optimum in the sense of Definition 4.15.*

Proof. Under Assumption 4.1(iii), the cache stores the exact objective value of every evaluated point, returned without re-evaluation on subsequent lookups; in particular, repeated access to a cached point does not alter its value. By hypothesis, every one-coordinate perturbation of h^* has been evaluated and none has a strictly smaller value than $f(h^*)$. Hence $f(h^*) \leq f(h)$ for every one-coordinate perturbation h of h^* , which is precisely Definition 4.15. \square

Proposition 4.16 is the primary algorithmic guarantee of DNR-Search. Although the result is elementary in form — if all one-coordinate alternatives have been tested and none improves, then the point is coordinate-wise locally optimal — its operational significance lies in the fact that the certificate is obtained *entirely from the algorithmic data structure*: the canonical cache accumulates evaluation results throughout the search and, after the final incumbent is reached, certifies local optimality through cache lookups alone, without any additional function evaluations. It is logically independent of the schedule-level pairwise adjacency coverage result (Proposition 4.8): the stopping certificate follows from the cache structure and Definition 4.15, not from any property of the permutation schedule.

Remark 4.17 (Scope and limitations of the certificate). Two limitations of the certificate should be stated clearly. First, Definition 4.15 and Proposition 4.16 concern *one-coordinate perturbations only*: the certificate does not exclude improvement by simultaneous changes in multiple coordinates, and a coordinate-wise local optimum need not be a global or multi-coordinate local optimum. Problems with strong variable interactions, where improvement requires changing two or more coordinates simultaneously, will not be detected by this certificate. Second, the certificate rests on Assumption 4.1(iv) (full domain feasibility): all $n(m - 1)$ one-variable alternatives of h^* must be evaluable. Problems with forbidden categorical combinations require a reformulation of the local-optimality notion relative to the feasible neighbourhood only; constrained categorical feasibility is left for future work.

Safety-net criterion (N_{stop}): the algorithm also stops after N_{stop} consecutive permutations without improvement, where $N_{\text{stop}} = n \cdot (m - 1)$. This ensures finite termination even if coverage is not achieved within the expected number of permutations.

Remark 4.18 (Certifying versus non-certifying termination). The two stopping criteria have different guarantees. If the algorithm terminates via the *primary coverage criterion* (`covered = true`), then Proposition 4.16 applies and h^* is a certified coordinate-wise categorical local optimum. If the algorithm terminates via the *safety-net criterion* ($\text{cnt} \geq N_{\text{stop}}$), then h^* is the best point found but no local optimality certificate is available; the incumbent may or may not be a coordinate-wise local optimum. In the numerical experiments of Section 6, the primary coverage criterion always fires first.

PROPOSITION 4.19 (Finite termination of DNR-Search). *Under Assumption 4.1, Algorithm 4.1 terminates in finite time.*

Proof. The argument rests on two observations.

(a) *Finitely many successful incumbent updates.* Every accepted step replaces h^* by a point with strictly smaller objective value (by the acceptance condition $f(\psi(h')) < f^*$). Since the domain $\{1, \dots, m\}^n$ is finite, the set of attainable objective values is finite, and no point is accepted as incumbent more than once. Hence the total number of successful updates across the entire run is bounded by $m^n - 1$.

(b) *Finite exhaustion of each permutation.* Within a single inner loop, α is multiplied by $\beta \in (0, 1)$ at each failure. After at most $\lceil \log_\beta(1.5/m) \rceil$ failures without improvement, $\alpha < 1.5/m$ and $r(\alpha) = 1$. By observation (a), any successful updates within the inner loop are finite in number. Once no rank-1 improvement exists, the poll fails and the inner loop exits.

Outer loop. Each inner loop terminates in finite time by (b). The outer loop then terminates because at least one of two criteria must eventually fire: either the primary coverage test (all $n(m-1)$ one-variable alternatives of the current incumbent are cached) or the safety-net counter $\text{cnt} \geq N_{\text{stop}}$. Both criteria are checked after every permutation, and the total number of unique evaluations is bounded by m^n . Since the cache prevents re-evaluation of any point, each new function evaluation corresponds to a previously unseen point; the finite domain therefore precludes infinite execution of the outer loop independently of which criterion fires. \square

Remark 4.20 (Coverage verification is free after convergence). After the optimal point is found, coverage verification proceeds entirely through cache lookups. All subsequent trial points have been previously evaluated; the permutation advances without any new function evaluation. In DFO settings where each evaluation may cost hours of computation, this property is essential: the computational cost of confirming local optimality is negligible relative to the cost of function evaluation.

The key feature of the framework is that exploration richness is guaranteed at the schedule level, while correctness is ensured at the algorithmic level through cache-based certification.

Algorithm 4.1 states the complete DNR-Search method.

Remark 4.21 (Computational cost). The cost per permutation is dominated by at most $2n$ cache lookups and at most $2n$ function evaluations. Generating each Sobol-rank permutation requires $O(m \log m)$ operations. After convergence, all subsequent permutations cost only cache lookups — zero function evaluations.

A detailed illustrative example ($n = 2$, $m = 4$) demonstrating the interaction between permutation rotation, caching, and coverage verification is provided in Appendix D.

5. Practical Implementation Guidelines.

Controlled disorder versus randomness. Although the rotating permutation schedule changes the neighborhood structure over time, it is important to distinguish this from stochastic exploration. The DNR mechanism introduces *controlled disorder*: every neighborhood configuration is uniquely determined by the evaluation count and the fixed parameters (s_0, ω) , so the entire sequence of evaluated points is reproducible from a fixed problem definition. Randomness, by contrast, generates adjacency diversity through stochastic perturbations that depend on pseudo-random generators and seeding. The DNR mechanism achieves the same structural diversity deterministically: no categorical pair is permanently excluded from rank-1 adjacency by the schedule, yet the entire sequence is reproducible. A detailed schedule comparison (cyclic-shift versus Sobol-rank, $m = 5$) is provided in Appendix A.

5.1. Choice of parameters. DNR-Search involves five interpretable parameters, all inherited from standard pattern search, summarized in Table 5.1. The base index s_0 selects the starting point in the Sobol sequence; it is fixed to $s_0 = 0$ by convention throughout and is not counted as a free parameter.

Variable-specific permutation indices. In problems with multiple categorical variables, applying the same permutation to all variables would cause their neighborhood structures to change synchronously, reducing diversity. To avoid this, each variable $i \in \{1, \dots, n\}$ uses an independent permutation index

$$(5.1) \quad s_i = s_0 + s + \omega \cdot i,$$

where $\omega > 0$ is the *variable offset*. This ensures that each variable draws from a different part of the Sobol sequence at every permutation step. For example, with $n = 3$, $s_0 = 0$, $\omega = 1000$, and $s = 1$, the indices are $s_1 = 1001$, $s_2 = 2001$, $s_3 = 3001$. The offset ω should be large enough to prevent overlap between variables' sequences; $\omega = 1000$ is the default and is sufficient in all cases tested. This design choice does not affect the validity of Proposition 4.8 or Proposition 4.16.

TABLE 5.1
Default parameters of DNR-Search. The base index $s_0 = 0$ is fixed by convention.

Parameter	Default	Role
α_0	0.5	Initial step size; gives starting rank $r_0 = \text{round}(m/2)$
β	0.5	Contraction factor; governs rank decrease on failure
γ	1.0	Expansion factor; $\gamma = 1$ means no expansion
ω	1000	Variable offset; ensures independent permutation schedules
N_{stop}	$n(m - 1)$	Safety-net bound; conservative, coverage always fires first

The computational cost is dominated by function evaluations. Generating a Sobol-rank permutation requires $O(m \log m)$ operations, and all candidate points are canonically encoded and cached via the pipeline of Definition 4.2–4.13, which is compatible with mixed normalized search spaces $[0, 1]^n$. After convergence, certification proceeds entirely through cache lookups with zero additional evaluations (Remark 4.20).

6. Numerical Experiments. This section presents three series of numerical experiments that assess the correctness, scaling behaviour, and comparative performance of DNR-Search against two natural baselines: integer-encoding search (IE) and random-neighbor selection (RNS). All experiments use the default parameters of Section 5.1: $\alpha_0 = 0.5$, $\beta = 0.5$, $\gamma = 1.0$, $s_0 = 0$, $\omega = 1000$, $N_{\text{stop}} = n(m - 1)$.

6.1. Experimental setup.

All-categorical representation.. To isolate the DNR mechanism from continuous search geometry, all benchmark problems are represented in a purely categorical form. For each variable i , a uniform grid of m values over $[l_i, u_i]$ is constructed with the grid point closest to x_i^* replaced by x_i^* exactly. A fixed non-monotone permutation (odd-even interleaving followed by a variable-index rotation) is then applied so that categorical adjacency does not coincide with physical adjacency. This deliberate decorrelation ensures that the search structure is genuinely categorical: any method that relies on an imposed integer order on the category indices cannot benefit from hidden numerical structure.

Benchmark problems.. Three standard continuous problems are used: sphere ($f(x) = \sum x_i^2$, $[-5, 5]^n$), Rosenbrock ($f(x) = \sum [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$, $[-2, 2]^n$), and Rastrigin ($f(x) = 10n + \sum [x_i^2 - 10 \cos(2\pi x_i)]$, $[-5.12, 5.12]^n$). These represent qualitatively different landscapes: unimodal convex, narrow-valley, and multimodal.

Baseline methods.. Two baselines are included to contextualise the DNR results. These baselines represent the two canonical alternatives identified in Section 3: imposed artificial geometry (IE) and stochastic exploration (RNS), respectively. Their inclusion is intentional: the experimental results directly test whether the limitations described in Sections 3.1 and 3.2 manifest empirically under a non-monotone categorical embedding.

Integer-encoding search (IE) maps categories to consecutive integers $\{1, \dots, m\}$ and performs opportunistic first-improvement coordinate sweeps with steps $\delta = \pm 1$ on the canonical indices. It terminates when a full sweep completes without improvement (*loc*), which certifies a local optimum in the induced integer neighborhood. IE is deterministic and uses a cache to avoid re-evaluation. It represents a common practical treatment of categorical variables in derivative-free optimization. Because the embedding is deliberately non-monotone, canonical adjacency has no relation to physical adjacency, so IE cannot benefit from any hidden numerical structure. To assess whether the observed IE gaps are due to a single unfavorable starting point rather than a structural limitation, a multi-start variant (IE-MS) is also run: 10 random initial points are used, and the best, median, and worst gap over the 10 runs are reported.

Random-neighbor selection (RNS) selects a random variable i and a random category $k' \neq h^*(i)$ at each iteration, accepting only on improvement. No cache or memory is maintained; the method is entirely memory-free. It is run 20 times with seeds $1, \dots, 20$ to characterise its statistical behaviour. RNS terminates after $N_{\text{stop}} = n(m - 1)$ consecutive evaluations without improvement — the same value used as DNR’s safety net — enabling a direct natural-stopping comparison. RNS serves as a stochastic lower bound: a baseline that avoids artificial structure but sacrifices determinism and memory.

Comparison modes.. Two comparison modes are used. *Mode A (natural stopping)* lets each method use its own stopping criterion: DNR by coverage certificate (*cov*) or safety net (*sft*); IE at its integer local optimum (*loc*); RNS after N_{stop} consecutive failures. *Mode B (fixed budget)* stops all three methods when $\text{eval.count} \geq B$, with $B = C \cdot n(m - 1) + 1$ and $C \in \{1, 2, 5\}$. The +1 accounts for the shared initial evaluation. Any method that reaches its own stopping criterion before exhausting B stops early; this is recorded in the Stop column.

Metrics.. For DNR in Mode A we report: unique evaluations (*Evals*), new evaluations after the last improvement (*Conf*, the certification cost), and stopping criterion (*Stop*). For RNS we report best, median, and worst gap over 20 runs; median blind

tail (*Tail*: evaluations after the best solution was found); and success rate (*Succ*: fraction of runs achieving gap $\leq 10^{-6}$).

6.2. Scalability in the number of categories. Table 6.1 reports the DNR-only results for the Rastrigin problem with $n = 3$ and $m \in \{10, 20, 50, 100, 200\}$.

TABLE 6.1

Series 1: scalability in m (Rastrigin, $n = 3$). After evals and After hits count unique evaluations and cache lookups after the final incumbent was first attained. Cache% is the fraction of post-solution activity handled by cache lookups.

m	Evals	Cache hits	Perms	After evals	After hits	Cache%	$f - f^*$	Stop
10	40	164	13	23	163	88	0.00	<i>cov</i>
20	104	439	25	42	384	90	0.00	<i>cov</i>
50	302	1 755	61	144	1 662	92	0.00	<i>cov</i>
100	585	4 496	129	267	4 218	94	0.00	<i>cov</i>
200	1 123	9 592	237	424	7 424	95	0.00	<i>cov</i>

DNR-Search finds the known optimum and certifies coordinate-wise local optimality in all cases, always via the coverage certificate. The post-solution cache fraction increases monotonically with m (88% to 95%), confirming that certification becomes increasingly cache-driven as the number of categories grows.

Tables 6.2 and 6.3 compare all three methods. Three observations stand out. First, IE terminates with 12–39 evaluations across all values of m , immediately at a local optimum of the integer neighborhood. Its gap grows from 0 at $m = 10$ to 75 at $m = 200$ while its evaluation count remains essentially constant — the signature of representation dependence: under the non-monotone embedding, integer adjacency becomes structurally irrelevant. A multi-start variant of IE (10 random initial points) is analysed in Section 6.5: random restarts sometimes reduce the gap but do not resolve the structural limitation, confirming that the main cause is representation-induced rather than initialization-dependent. Second, under Mode A, RNS achieves median gap zero for $m \geq 20$ but with non-trivial worst-case gaps, and its blind tail equals exactly $N_{\text{stop}} = n(m-1)$ in every run — confirming that RNS always exhausts its stopping horizon, continuing to sample without detecting that the best solution has already been attained. Third, under Mode B with $C = 2$, DNR certifies local optimality in 4 out of 5 configurations (all except $m = 50$, where the budget is marginally insufficient), while neither IE nor RNS provides any certification mechanism.

TABLE 6.2

Series 1: scalability in m (Rastrigin, $n = 3$) — natural stopping

m	DNR-Search				IE		RNS (20 runs)				
	Gap	Evals	Conf	Stop	Gap	Evals	Best	Median	Worst	Tail [†]	Succ [‡]
10	0	40	23	<i>cov</i>	0	12	0	6.25	25	27	50%
20	0	104	42	<i>cov</i>	18	39	0	0	6	57	75%
50	0	302	144	<i>cov</i>	61	11	0	0	3.15	147	70%
100	0	585	267	<i>cov</i>	66.4	22	0	0	1.05	297	65%
200	0	1123	424	<i>cov</i>	74.7	18	0	0	1	597	55%

[†] Median blind tail: evaluations after best point found (RNS, Mode A).

[‡] Fraction of 20 runs achieving gap $\leq 10^{-6}$.

DNR “Conf” = new evaluations after last improvement (certification cost).

DNR “Stop”: *cov* = coverage certificate; *sft* = safety net.

Tables 6.3–6.9 report the fixed-budget comparison at $C = 2$; full results for $C \in \{1, 2, 5\}$ across all three series are available in the supplementary material.

TABLE 6.3
Series 1: scalability in m (Rastrigin, $n = 3$) — fixed budget $B = 2n(m - 1) + 1$

m	B	DNR-Search			IE			RNS (20 runs)			
		Gap	Evals	Stop	Gap	Evals	Stop	Best	Median	Worst	Succ [†]
10	55	0	40	<i>cov</i>	0	12	<i>loc</i>	0	6.25	25	50%
20	115	0	104	<i>cov</i>	18	39	<i>loc</i>	0	0	6	65%
50	295	0	295	<i>bgt</i>	61	11	<i>loc</i>	0	0	3.15	65%
100	595	0	585	<i>cov</i>	66.4	22	<i>loc</i>	0	0	1.05	60%
200	1195	0	1123	<i>cov</i>	74.7	18	<i>loc</i>	0	0	1	55%

[†] Fraction of 20 runs achieving gap $\leq 10^{-6}$ within budget B .

Stop codes: *cov* = coverage certificate; *loc* = integer local optimum; *bgt* = budget exhausted; *sft* = safety net.

Remark 6.1 (Non-monotone coverage curves). When the incumbent improves, coverage may temporarily decrease: the new incumbent has a different set of one-variable neighbours, most of which have not yet been evaluated. This is not a deficiency of the method but the expected consequence of the separation between canonical identity and neighbourhood structure (Principle P2).

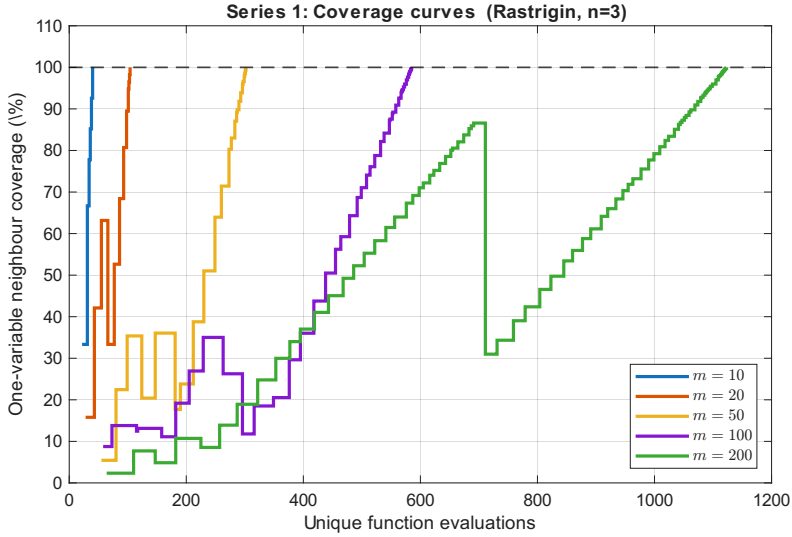


FIG. 6.1. Series 1: one-variable neighbour coverage (%) as a function of unique function evaluations, for $m \in \{10, 20, 50, 100, 200\}$ (Rastrigin, $n = 3$). Coverage drops occur when the incumbent improves and its new neighbours have not yet been evaluated (*Remark 6.1*).

6.3. Scalability in the number of variables. Table 6.4 reports the DNR-only results for Rastrigin with $m = 20$ and $n \in \{2, 3, 5, 10, 15\}$.

The permutation count remains nearly constant (25–29) across all dimensions while evaluations grow approximately linearly with n , suggesting that the growth in cost comes primarily from the larger number of one-variable alternatives to certify

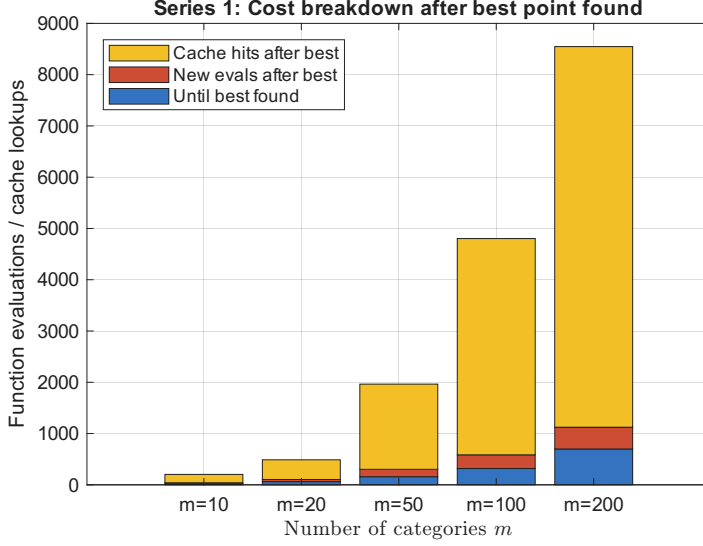


FIG. 6.2. *Series 1: cost breakdown after the final incumbent is first attained (Rastrigin, $n = 3$). Blue: evaluations until best found; red: new evaluations after that point; yellow: cache lookups for coverage verification. The dominant post-solution cost is cache-based and grows with m , confirming Remark 4.20.*

TABLE 6.4

Series 2: scalability in n (Rastrigin, $m = 20$). All runs stop via the coverage certificate with zero gap.

n	$n(m - 1)$	Evals	Cache hits	Perms	Stop
2	38	65	300	25	<i>cov</i>
3	57	104	439	25	<i>cov</i>
5	95	231	698	25	<i>cov</i>
10	190	615	1 458	27	<i>cov</i>
15	285	989	2 362	29	<i>cov</i>

within each permutation rather than from a need for more permutation changes. Tables 6.5 and 6.6 show the three-method comparison. The picture is consistent with Series 1: IE terminates immediately at a local optimum with gap growing roughly linearly with n ; RNS median gap is zero for small n but rises at intermediate dimensions; under Mode B with $C = 5$, DNR certifies in all cases.

Figure 6.3 summarises the scaling behaviour of DNR-Search under natural stopping. Panel (a) shows total evaluations and certification cost as functions of m ; panel (b) shows the same quantities as functions of n . In both panels the certification cost (dashed) remains a stable fraction of the total, confirming that the ratio of certification overhead to total cost does not grow with problem size.

Figure 6.4 compares all three methods under fixed budget for two representative configurations. IE gap is flat across all values of C because IE terminates at its integer local optimum in far fewer evaluations than any budget B . This invariance with respect to the budget confirms that the limitation is structural (representation-induced), rather than due to insufficient computational effort. DNR reaches zero gap at $C = 2$ in panel (a) and at $C = 5$ in panel (b); RNS median follows a similar trajectory in

TABLE 6.5
Series 2: scalability in n (Rastrigin, $m = 20$) — natural stopping

n	DNR-Search				IE		RNS (20 runs)				
	Gap	Evals	Conf	Stop	Gap	Evals	Best	Median	Worst	Tail [†]	Succ [‡]
2	0	65	23	<i>cov</i>	12	22	0	0	12	38	75%
3	0	104	42	<i>cov</i>	18	39	0	0	6	57	75%
5	0	231	84	<i>cov</i>	30	66	0	6	18	95	45%
10	0	615	179	<i>cov</i>	60	224	0	0	12	190	60%
15	0	989	282	<i>cov</i>	84	325	0	0	18	285	55%

[†] Median blind tail: evaluations after best point found (RNS, Mode A).

[‡] Fraction of 20 runs achieving gap $\leq 10^{-6}$.

DNR “Conf” = new evaluations after last improvement (certification cost).

DNR “Stop”: *cov* = coverage certificate; *sft* = safety net.

TABLE 6.6
Series 2: scalability in n (Rastrigin, $m = 20$) — fixed budget $B = 2n(m - 1) + 1$

n	B	DNR-Search			IE			RNS (20 runs)			
		Gap	Evals	Stop	Gap	Evals	Stop	Best	Median	Worst	Succ [†]
2	77	0	65	<i>cov</i>	12	22	<i>loc</i>	0	0	12	75%
3	115	0	104	<i>cov</i>	18	39	<i>loc</i>	0	0	6	65%
5	191	0	191	<i>bgt</i>	30	66	<i>loc</i>	0	6	18	40%
10	381	6	381	<i>bgt</i>	60	224	<i>loc</i>	0	6	18	35%
15	571	18	571	<i>bgt</i>	84	325	<i>loc</i>	0	12	24	25%

[†] Fraction of 20 runs achieving gap $\leq 10^{-6}$ within budget B .

Stop codes: *cov* = coverage certificate; *loc* = integer local optimum; *bgt* = budget exhausted; *sft* = safety net.

terms of solution quality, but lacks any mechanism to certify local optimality.

6.4. Behaviour across problem structures. Tables 6.7, 6.8, and 6.9 compare all three methods across Sphere, Rosenbrock, and Rastrigin with $n = 3$, $m = 20$.

TABLE 6.7
Series 3: behaviour across problem structures ($n = 3$, $m = 20$). All DNR runs reach zero gap; all stop via the coverage certificate.

Problem	Evals	Cache hits	Perms	After evals	After hits	Stop
Sphere	70	468	25	53	463	<i>cov</i>
Rosenbrock	68	258	15	52	254	<i>cov</i>
Rastrigin	104	439	25	42	384	<i>cov</i>

Two additional observations emerge. Under natural stopping, IE achieves zero gap for Rosenbrock but fails for Sphere (gap = 0.21) and Rastrigin (gap = 18), demonstrating label dependence: performance is sensitive to the arbitrary encoding. For RNS, Rosenbrock yields the largest worst-case gap (7.34) despite a moderate median, illustrating the statistical variability intrinsic to memory-free random search. DNR achieves zero gap in all three cases under both modes and certifies coordinate-wise local optimality in each.

6.5. Multi-start integer-encoding. To assess whether the IE gaps in Sections 6.2–6.4 are merely an artifact of the fixed central starting point, IE is re-run

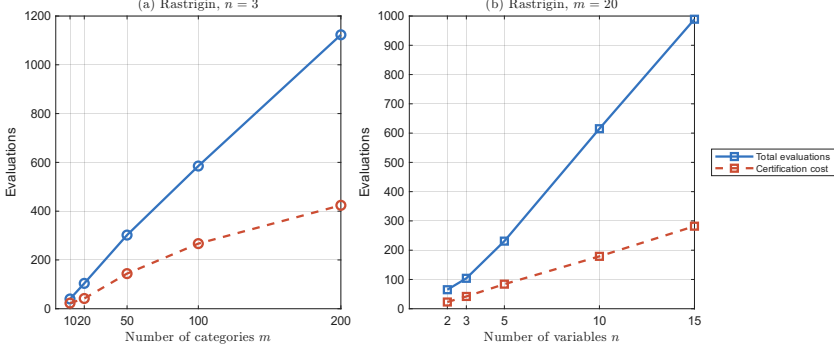


FIG. 6.3. *Natural-stopping scaling of DNR-Search. (a) Total evaluations (solid) and certification cost after last improvement (dashed) as a function of the number of categories m (Rastrigin, $n = 3$). (b) Same quantities as a function of the number of variables n (Rastrigin, $m = 20$). In both panels, evaluations grow approximately linearly in the problem parameter while the certification cost remains a stable fraction of the total.*

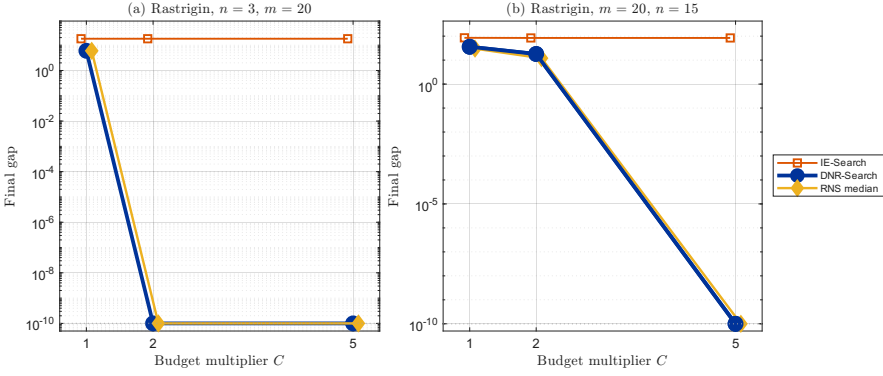


FIG. 6.4. *Fixed-budget comparison of DNR-Search, IE-Search, and RNS median gap as a function of budget multiplier C , where $B = Cn(m - 1) + 1$. (a) Rastrigin, $n = 3$, $m = 20$. (b) Rastrigin, $m = 20$, $n = 15$. Values at the lower axis limit (10^{-10}) correspond to gap = 0. IE gap is constant across C because IE always terminates at its integer local optimum well below any budget B .*

from 10 random initial points (uniformly sampled from $\{1, \dots, m\}^n$, seed fixed for reproducibility). Table 6.10 reports a representative subset of results.

Multi-start improves the best-case IE gap in several configurations (e.g., $m = 20$: from 18 to 6.0; $m = 100$: from 66.4 to 9.75), confirming that initialization does have some effect. However, the improvement is limited and inconsistent: the best gap over 10 starts remains substantially positive for $m \geq 20$ and grows markedly with n (best = 77.6 at $n = 15$), and the median gap stays large throughout. This suggests that the main limitation of IE is structural, stemming from the imposed integer neighborhood under the non-monotone categorical embedding, rather than being solely an artifact of initialization. This reinforces the interpretation that the observed performance gap is not due to insufficient exploration, but to the structural mismatch between imposed integer neighborhoods and categorical problem geometry.

Interestingly, on Rosenbrock the fixed central start attains zero gap, whereas no random restart does (best = 0.85, median = 4.50, worst = 809). This illustrates that

TABLE 6.8

Series 3: behaviour across problem structures ($n = 3$, $m = 20$) — natural stopping

Problem	DNR-Search				IE		RNS (20 runs)				
	Gap	Evals	Conf	Stop	Gap	Evals	Best	Median	Worst	Tail [†]	Succ [‡]
Sphere	0	70	53	<i>cov</i>	0.21	21	0	0	0.62	57	50%
Rosenbrock	0	68	52	<i>cov</i>	0	12	0	1.25	7.34	57	50%
Rastrigin	0	104	42	<i>cov</i>	18	39	0	0	6	57	75%

[†] Median blind tail: evaluations after best point found (RNS, Mode A).

[‡] Fraction of 20 runs achieving gap $\leq 10^{-6}$.

DNR “Conf” = new evaluations after last improvement (certification cost).

DNR “Stop”: *cov* = coverage certificate; *sft* = safety net.

TABLE 6.9

Series 3: problem structures ($n = 3$, $m = 20$) — fixed budget $B = 2n(m - 1) + 1$

Problem	B	DNR-Search			IE			RNS (20 runs)			
		Gap	Evals	Stop	Gap	Evals	Stop	Best	Median	Worst	Succ [†]
Sphere	115	0	70	<i>cov</i>	0.21	21	<i>loc</i>	0	0	0.76	50%
Rosenbrock	115	0	68	<i>cov</i>	0	12	<i>loc</i>	0	1.25	7.34	50%
Rastrigin	115	0	104	<i>cov</i>	18	39	<i>loc</i>	0	0	6	65%

[†] Fraction of 20 runs achieving gap $\leq 10^{-6}$ within budget B .

Stop codes: *cov* = coverage certificate; *loc* = integer local optimum; *bgt* = budget exhausted; *sft* = safety net.

IE is highly initialization-dependent: favorable performance may occur for specific starting points but is not robust across starts. DNR-Search achieves zero gap from the central starting point in all configurations considered here, without relying on any favorable initialization.

TABLE 6.10

Multi-start IE (IE-MS, 10 random initial points, seed 42): representative configurations from all three series (natural stopping). Multi-start occasionally reduces the IE gap but does not resolve the structural limitation; DNR achieves $f - f^* = 0$ in every case.

Series	Config.	DNR	IE (fixed)	IE-MS (10 starts)	
				Best	Median
S1 ($n = 3$)	$m = 20$	0	18	6.0	16.5
	$m = 100$	0	66.4	9.75	25.6
	$m = 200$	0	74.7	11.5	29.8
S2 ($m = 20$)	$n = 3$	0	18	6.0	15.0
	$n = 10$	0	60	36	54.9
	$n = 15$	0	84	77.6	103
S3 ($n = 3$, $m = 20$)	Sphere	0	0.21	0	0.14
	Rosenbrock	0	0	0.85	4.50
	Rastrigin	0	18	6.0	18

All gaps are $f - f^*$. Full results (all 15 configurations, best/median/worst) in `results_ie_multistart.mat`.

6.6. Summary. DNR-Search certifies coordinate-wise local optimality in all 15 configurations tested (3 series \times 5 settings), always via the primary coverage certifi-

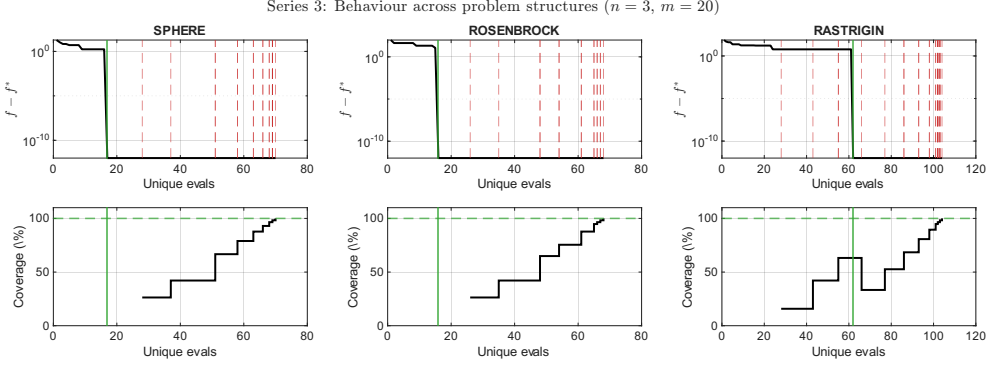


FIG. 6.5. *Series 3: convergence (upper panels, log scale) and one-variable neighbour coverage (lower panels) for Sphere, Rosenbrock, and Rastrigin ($n = 3, m = 20$). Red dashed lines mark permutation changes; the green vertical line marks the evaluation at which the final incumbent was first attained. After that point, the objective value is constant while coverage is completed through cache lookups.*

cate; the safety net never fires, and post-convergence certification exceeds 88% cache-based in all cases. On the constructed benchmarks, the certified coordinate-wise local optimum coincides with the known global optimum in all tested configurations — a consequence of the specific embedding design, not a general guarantee of global optimality. The IE performance gap persists across all 10 random restarts (Table 6.10), confirming a structural limitation rather than an initialization artifact; this is consistent with the invariance of IE under budget increases (Mode B). RNS provides no certification mechanism and achieves 0% success rates at $n = 10$ under $C = 1$. Under Mode B with $C = 2$, DNR certifies in 9 of 15 configurations while neither IE nor RNS certifies in any (Figure 6.4). These results confirm that the separation principle (P2), together with determinism (P4) and controlled exploration (P5), is sufficient to achieve a deterministic and certifiable framework for coordinate-wise optimization over categorical domains — a property not exhibited by either IE or RNS.

7. Conclusions. Categorical variables have long resisted principled treatment in direct-search optimization: existing approaches either impose an artificial ordering that biases the search, or rely on stochastic operators that sacrifice reproducibility. This work resolves both limitations within a single deterministic framework — neighborhood rotation — that replaces artificial ordering and stochastic exploration with a principled, rotating neighborhood structure operating entirely in the nominal categorical space.

The present contribution should be understood as a *finite-domain algorithmic foundations* paper, not a continuous convergence theory paper. The main guarantees established are: finite termination under a deterministic stopping rule, and a cache-based certificate of coordinate-wise categorical local optimality. These are provably correct and operationally meaningful on finite categorical product spaces, but they are not analogues of the MADS/GPS asymptotic convergence theory for continuous search. That distinction should be kept clearly in view when interpreting the paper’s scope.

The resulting algorithm, DNR-Search, is built on two logically independent guarantees. The exploration component (Proposition 4.8) establishes that any dense per-

mutation schedule eventually exposes every pair of categories as rank-1 neighbours, ensuring that no categorical transition is systematically excluded from the exploration process. The certification component (Proposition 4.16) establishes that, once all one-variable alternatives of the current incumbent are in the cache without improvement, a coordinate-wise categorical local optimum has been certified. The independence of these two results is a structural feature of the framework: the correctness of the stopping certificate does not depend on the exploration guarantee.

The algorithm is parameterised by five interpretable quantities inherited from standard pattern search (with the base index $s_0 = 0$ fixed by convention). Its computational overhead is negligible relative to function evaluation cost, and local optimality certification after convergence proceeds entirely through cache lookups — incurring zero additional evaluations. These properties make DNR-Search compatible with the requirements of black-box and simulation-based problems where each evaluation may cost hours of computation.

Several directions for future work follow naturally from this foundation. The companion work [?] provides the structural metric foundation for the DNR mechanism: it formalises mixed search spaces as weighted Cartesian products of metric spaces, proves a tight coverage bound of $\lceil (k-1)/2 \rceil$ orderings for full categorical adjacency coverage, and establishes that ordering rotation preserves the topology of the search space through uniform metric equivalence. That framework also introduces a hierarchy of local optimality—instantaneous, robust, and categorical—that contextualises the coordinate-wise certificate proved here within a broader geometric theory of mixed-variable optimisation. The integration of DNR-Search into mixed continuous-categorical frameworks is the most immediate algorithmic extension: the DNR mechanism provides a deterministic categorical neighborhood component that could be embedded within MADS-based solvers such as CatMADS [1] or combined with CMA-ES-based methods such as CatCMAwM [12], replacing their stochastic or distance-based categorical polling with a fully deterministic alternative. Developing a formal convergence theory for the coordinate search variant — establishing conditions under which the algorithm identifies a global optimum or escapes coordinate-wise local optima — is a deeper open problem. A systematic numerical comparison against CatMADS [1] and CatCMAwM [12] on standard mixed-variable benchmarks, including the Cat-Suite problems of [1], would further characterise the practical scope of DNR-Search relative to state-of-the-art methods; this is a priority direction for future work.

The additional multi-start analysis of the integer-encoding baseline further confirms that the observed performance gap is not merely due to initialization, but reflects a structural limitation of imposed integer neighborhoods under non-monotone categorical embeddings.

The present work demonstrates that a finite deterministic framework for coordinate-wise local optimality certification over categorical domains can be achieved without imposing artificial structure or relying on randomness. The mechanism introduced here — separating canonical identity from neighborhood exploration and certifying optimality through cache-based verification — constitutes a reusable design principle applicable to any direct-search framework that must handle qualitative decision variables in a deterministic and reproducible manner.

Acknowledgments. The MATLAB implementation of DNR-Search and all numerical results reported in this paper will be made publicly available on GitHub upon acceptance for publication.

REFERENCES

- [1] C. AUDET, Y. DIOUANE, E. HALLÉ-HANNAN, S. LE DIGABEL, AND C. TRIBES, *CatMADS: Mesh adaptive direct search for constrained blackbox optimization with categorical variables*, arXiv preprint arXiv:2506.06937, (2025).
- [2] C. AUDET AND W. HARE, *Derivative-Free and Blackbox Optimization*, Springer Series in Operations Research and Financial Engineering, Springer, 2017.
- [3] ———, *A survey on direct search methods for blackbox optimization and their applications*, in Mathematics Without Boundaries, Springer, 2019, pp. 31–56.
- [4] C. AUDET AND S. LE DIGABEL, *A survey on direct search methods for blackbox optimization and their applications*, in Mathematics Without Boundaries, T. M. Rassias and P. Pardalos, eds., Springer, 2019, pp. 31–56.
- [5] A. R. CONN, K. SCHEINBERG, AND L. N. VICENTE, *Introduction to Derivative-Free Optimization*, MPS-SIAM Series on Optimization, SIAM, 2009.
- [6] A. L. CUSTÓDIO AND L. N. VICENTE, *Analysis of direct searches for discontinuous functions*, Mathematical Programming, 133 (2012), pp. 299–325.
- [7] K. DEB, A. PRATAP, S. AGARWAL, AND T. MEYARIVAN, *A fast and elitist multiobjective genetic algorithm: NSGA-II*, IEEE Transactions on Evolutionary Computation, 6 (2002), pp. 182–197.
- [8] E. C. GARRIDO-MERCHÁN AND D. HERNÁNDEZ-LOBATO, *Dealing with categorical and integer-valued variables in Bayesian optimization with Gaussian processes*, Neurocomputing, 380 (2020), pp. 20–35.
- [9] D. E. GOLDBERG, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.
- [10] M. HALSTRUP, *Black-box optimization of mixed discrete-continuous optimization problems*, PhD thesis, TU Dortmund University, 2016.
- [11] J. H. HALTON, *On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals*, Numerische Mathematik, 2 (1960), pp. 84–90.
- [12] R. HAMANO, M. NOMURA, S. SAITO, K. UCHIDA, AND S. SHIRAKAWA, *CatCMA with margin: Stochastic optimization for continuous, integer, and categorical variables*, in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '25), New York, NY, USA, 2025, ACM, pp. 737–745.
- [13] R. HAMANO, S. SAITO, M. NOMURA, K. UCHIDA, AND S. SHIRAKAWA, *CatCMA: Stochastic optimization for mixed-category problems*, in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '24), New York, NY, USA, 2024, ACM, pp. 656–664.
- [14] F. HUTTER, H. H. HOOS, AND K. LEYTON-BROWN, *Sequential model-based optimization for general algorithm configuration*, Journal of Artificial Intelligence Research, 41 (2011), pp. 507–547.
- [15] S. JOE AND F. Y. KUO, *Remark on algorithm 659: Implementing Sobol’s quasirandom sequence generator*, ACM Transactions on Mathematical Software, 29 (2003), pp. 49–57.
- [16] S. KIRKPATRICK, C. D. GELATT, AND M. P. VECCHI, *Optimization by simulated annealing*, Science, 220 (1983), pp. 671–680.
- [17] T. G. KOLDA, R. M. LEWIS, AND V. TORCZON, *Optimization by direct search: New perspectives on some classical and modern methods*, SIAM Review, 45 (2003), pp. 385–482.
- [18] G. LIUZZI, S. LUCIDI, AND F. RINALDI, *Derivative-free methods for bound constrained mixed-integer optimization*, Computational Optimization and Applications, 53 (2012), pp. 505–526.
- [19] J. J. MORÉ AND S. M. WILD, *Benchmarking derivative-free optimization algorithms*, SIAM Journal on Optimization, 20 (2009), pp. 172–191.
- [20] H. NIEDERREITER, *Random Number Generation and Quasi-Monte Carlo Methods*, vol. 63 of CBMS-NSF Regional Conference Series in Applied Mathematics, SIAM, 1992.
- [21] J. PELAMATTI, L. BREVAULT, M. BALESSENT, E.-G. TALBI, AND Y. GUÉRIN, *Efficient global optimization of constrained mixed variable problems*, Journal of Global Optimization, 73 (2019), pp. 583–613.
- [22] Y. ZHANG, S. TAO, W. CHEN, AND D. W. APLEY, *A latent variable approach to Gaussian process modeling with qualitative and quantitative factors*, in Advances in Neural Information Processing Systems, vol. 33, 2020.

Appendix A. Neighborhood evolution table.

Table A.1 illustrates the difference between the cyclic-shift and Sobol-rank schedules for $m = 5$ categories over four permutation steps.

TABLE A.1

Evolution of rank-1 neighborhoods for $m = 5$ categories. Top: cyclic-shift schedule — neighborhoods are invariant under all shifts, confirming that categories non-adjacent in $p^{(0)}$ are never rank-1 neighbors. Bottom: Sobol-rank schedule — neighborhoods vary across steps, with all pairs eventually covered (Proposition 4.8).

Step	Permutation $p^{(s)}$	$\mathcal{N}(1)$	$\mathcal{N}(2)$	$\mathcal{N}(3)$	$\mathcal{N}(4)$	$\mathcal{N}(5)$
<i>Cyclic-shift schedule</i>						
$s = 0$	(1, 2, 3, 4, 5)	{2, 5}	{1, 3}	{2, 4}	{3, 5}	{1, 4}
$s = 1$	(2, 3, 4, 5, 1)	{2, 5}	{1, 3}	{2, 4}	{3, 5}	{1, 4}
$s = 2$	(3, 4, 5, 1, 2)	{2, 5}	{1, 3}	{2, 4}	{3, 5}	{1, 4}
$s = 3$	(4, 5, 1, 2, 3)	{2, 5}	{1, 3}	{2, 4}	{3, 5}	{1, 4}
<i>Sobol-rank schedule (Definition 4.6)</i>						
$s = 0$	(1, 2, 3, 4, 5)	{2, 5}	{1, 3}	{2, 4}	{3, 5}	{1, 4}
$s = 1$	(2, 3, 4, 1, 5)	{4, 5}	{3, 5}	{2, 4}	{1, 3}	{1, 2}
$s = 2$	(1, 5, 2, 3, 4)	{4, 5}	{3, 5}	{2, 4}	{1, 3}	{1, 2}
$s = 3$	(1, 2, 5, 3, 4)	{2, 4}	{1, 5}	{4, 5}	{1, 3}	{2, 3}

Appendix B. Variable type taxonomy.

This appendix provides standard definitions and algorithmic treatments for the three variable types that, unlike categorical variables, admit natural geometric structure.

Continuous variables.. A continuous variable takes values in $[l_i, u_i] \subset \mathbb{R}$ and is normalized via $y_i = (x_i - l_i)/(u_i - l_i) \in [0, 1]$. Continuous variables are ordered, metrizable, and refinable [5, 17, 19].

Integer variables.. An integer variable takes values in $\{l_i, \dots, u_i\} \subset \mathbb{Z}$ and is handled by relaxing to \mathbb{R} and rounding: $x_i = \text{round}(\tilde{x}_i)$ [18]. Integer variables are ordered and metrizable but have limited refinability once the step size falls below 1.

Discrete numerical variables.. A discrete numerical variable takes values in a finite ordered set $\{v_1 < v_2 < \dots < v_{m_i}\}$ and is handled via nearest-neighbor projection $x_i = \arg \min_{v \in \mathcal{V}_i} |\tilde{x}_i - v|$ [2]. Refinability is limited by the minimum spacing $\min_j (v_{j+1} - v_j)$.

All three types can be treated within a unified geometric framework, in contrast to categorical variables (Section 2).

Appendix C. Additional encoding strategies.

Two further encoding strategies for categorical variables appear in the literature but are less compatible with geometric DFO frameworks.

One-hot encoding.. One-hot encoding represents a categorical variable with m categories as a binary vector of dimension m ,

$$(C.1) \quad c_k \mapsto e_k = (0, \dots, 0, \underbrace{1}_{k\text{-th}}, 0, \dots, 0) \in \{0, 1\}^m.$$

This avoids imposing an explicit order among categories. Under the Hamming distance, any two distinct one-hot vectors satisfy $d_H(e_j, e_k) = 2$ for $j \neq k$. One-hot encoding is standard in machine learning and appears in surrogate-based optimization [8, 22]. However, it is problematic for geometric direct-search methods: with q categorical variables of sizes m_i , the representation adds $\sum_{i=1}^q (m_i - 1)$ degrees of freedom; the simplex constraints $\sum_j e_{ij} = 1$ introduce equalities; continuous relax-

ations $e_{ij} \in [0, 1]$ yield infeasible intermediate points; and Euclidean geometry in the embedded space does not reflect meaningful categorical similarity.

Latent variable approaches.. Bayesian optimization and related surrogate-based methods have proposed latent embeddings that map categories to continuous latent coordinates [22, 8]. While effective in surrogate-driven settings, these approaches are less suitable for direct-search algorithms: the optimization trajectory depends on a learned representation, training procedures often involve stochastic optimization, and the embedding must be updated as the search progresses, introducing significant overhead.

Appendix D. Illustrative example.

We illustrate the behaviour of DNR-Search on a simple categorical problem with $n = 2$ variables and $m = 4$ categories per variable. Each variable takes values in $\{1, 2, 3, 4\}$, and we consider the objective function

$$(D.1) \quad f(h) = (h_1 - 2)^2 + (h_2 - 3)^2,$$

which attains its minimum at $h^* = (2, 3)$ with $f(h^*) = 0$. The safety-net parameter is $N_{\text{stop}} = 2 \times 3 = 6$.

Initial permutation.. With $s_0 = 0$, $\omega = 1000$, and permutation step $s = 0$, the Sobol-rank construction yields permutations $p_1^{(0)}$ and $p_2^{(0)}$ for variables 1 and 2. For concreteness, suppose $p_1^{(0)} = (1, 3, 4, 2)$ and $p_2^{(0)} = (2, 4, 1, 3)$. Under $p_1^{(0)}$, category $k = 1$ is at position $j = 1$, so its rank-1 neighbours are $\mathcal{N}_{p_1^{(0)}}(1) = \{p_4, p_2\} = \{2, 3\}$.

Initial point and poll.. Starting from $h = (1, 1)$ with $f(1, 1) = 1 + 4 = 5$, $\alpha = 0.5$, and $r = 2$:

- Variable 1 forward: $h = (3, 1)$, $f = 1 + 4 = 5$ — no improvement.
- Variable 1 backward: $h = (4, 1)$, $f = 4 + 4 = 8$ — no improvement.
- Variable 2 forward: $h = (1, 4)$, $f = 1 + 1 = 2$ — improvement. Accept; $h \leftarrow (1, 4)$, $\alpha \leftarrow \min(1 \cdot 0.5, 1) = 0.5$.

After further polls with decreasing α , the algorithm reaches $h = (2, 3)$ with $f = 0$ within permutation $s = 0$ or $s = 1$.

Permutation change.. When $r = 1$ and no improving neighbour is found at the current incumbent, the permutation advances: $s \leftarrow s + 1$. The new permutations $p_1^{(1000+1)}$ and $p_2^{(2000+1)}$ induce different rank-1 adjacencies: category $k = 2$ may have neighbours $\{1, 4\}$ under $p^{(0)}$ and neighbours $\{3, 4\}$ under $p^{(1)}$. This is the DNR rotation in action: the same categorical value has different rank-1 neighbours at each permutation step.

Cache and coverage.. Every evaluated point is stored in the canonical cache \mathcal{C} . Once the incumbent is $h^* = (2, 3)$, the coverage verification checks: for variable 1, are all of $\{(1, 3), (3, 3), (4, 3)\}$ in \mathcal{C} ? For variable 2, are all of $\{(2, 1), (2, 2), (2, 4)\}$ in \mathcal{C} ? When all six one-variable alternatives are present, the algorithm terminates with the certificate: $h^* = (2, 3)$ is a coordinate-wise categorical local optimum. After the optimal incumbent is found, coverage verification involves at most $n(m - 1) = 6$ cache entries; no new objective evaluation is needed. In the numerical experiments (Section 6), post-solution activity exceeds 90% cache-based for $m \geq 10$.

Algorithm 4.1 DNR-Search: Deterministic Neighborhood Rotation coordinate search

Require: Initial point $h^{(0)}$; parameters $\alpha_0 \in (0, 1]$, $\beta \in (0, 1)$, $\gamma \geq 1$, $s_0 \geq 0$, $\omega > 0$,

$N_{\text{stop}} = n(m - 1)$

Convention: variables polled in order $i = 1, \dots, n$; for each variable h_i^+ tested before h_i^- ; if $h_i^+ = h_i^-$ (occurs when m even and $r = m/2$), only one trial is generated; only strict improvement ($f < f^*$) is accepted.

```

1: Evaluate  $f_0 = f(\psi(h^{(0)}))$ ; cache  $(h^{(0)}, f_0)$ ; set  $h^* \leftarrow h^{(0)}$ ,  $f^* \leftarrow f_0$ 
2: Set  $s \leftarrow s_0$ , cnt  $\leftarrow 0$ , covered  $\leftarrow$  false
3: while covered = false and cnt <  $N_{\text{stop}}$  do
4:   Generate  $p_i^{(s)}$  for each variable  $i$  via Sobol-rank (Def. 4.6)
5:   Set  $\alpha \leftarrow \alpha_0$ , improved  $\leftarrow$  false, exhausted  $\leftarrow$  false  $\triangleright \alpha$  resets to  $\alpha_0$  at each new
   permutation
6:   while exhausted = false do
7:      $r \leftarrow \max(1, \min(\lfloor m/2 \rfloor, \text{round}(\alpha m)))$ ; poll_success  $\leftarrow$  false
8:     for each variable  $i = 1, \dots, n$  do
9:       Compute rank- $r$  neighbors  $h_i^+, h_i^-$  under  $p_i^{(s)}$  (Def. 4.3, Eqs. (4.2)–(4.3))
10:      for each trial  $h' \in \{h_i^+, h_i^-\}$  do
11:        Look up  $h'$  in  $\mathcal{C}$ ; if absent, evaluate  $f(\psi(h'))$  and cache it
12:        if  $f(\psi(h')) < f^*$  then
13:           $h^* \leftarrow h'$ ;  $f^* \leftarrow f(\psi(h'))$ ; poll_success  $\leftarrow$  true; improved  $\leftarrow$  true
14:          break
15:        end if
16:      end for
17:      if poll_success = true then
18:        break
19:      end if
20:    end for
21:    if poll_success = true then
22:       $\alpha \leftarrow \min(\gamma\alpha, 1)$ 
23:    else
24:       $\alpha \leftarrow \beta\alpha$ 
25:      if  $r = 1$  then
26:        exhausted  $\leftarrow$  true
27:      end if
28:    end if
29:  end while
30:   $s \leftarrow s + 1$ 
31:  if improved = false then
32:    cnt  $\leftarrow$  cnt + 1
33:  else
34:    cnt  $\leftarrow 0$ 
35:  end if
36:  covered  $\leftarrow$  [all  $n(m - 1)$  one-variable alternatives of  $h^*$  are in  $\mathcal{C}$ ]
37: end while return  $h^*, f^*$ 

```
