Petroleum Science 23 (2026) 7309–7320

Contents lists available at ScienceDirect

Petroleum Science

journal homepage: www.keaipublishing.com/en/journals/petroleum-science

Original Paper

Constrained multi-objective well-pattern optimization for mature oilfields: A rule-aware, KD-tree-accelerated framework

Wan-Ming Xiong [a,b], Bei-Bei Xing [c], Lian-Bin Zhong [c], Guang Wang [c], Xiao-Long Chen [b], Long Chen [b], Rui Zhang [b], Yu-Jie Tian [b], Rui Huang [b], Hao-Ran Cheng [a,b,*]

[a] School of Chemical Engineering, Qingdao University of Science & Technology, Qingdao, 266042, Shandong, China
[b] Sichuan Energy Internet Research Institute, Tsinghua University, Chengdu, 610213, Sichuan, China
[c] Enhanced Oil Recovery Project Management Department, Daqing Oilfield Co., Ltd., Daqing, 163458, Heilongjiang, China

ARTICLE INFO

Article history:
Received 14 November 2025
Received in revised form
22 February 2026
Accepted 5 August 2026
Available online 8 August 2026

Edited by Xi Zhang

Keywords:
Well-pattern optimization
Multi-objective optimization
Mature oilfields
KD-tree
Differential evolution

ABSTRACT

Mature oilfield development requires the joint optimization of new well geometry with existing legacy well networks subject to operational constraints. However, existing manual workflows are time-consuming, subjective, and difficult to reproduce. This research presents an automated, rule-aware optimization framework that integrates domain expertise with computational geometry. The framework employs three solvers—differential evolution (DE), particle swarm optimization (PSO), and sequential least squares programming (SLSQP)—to generate reproducible and auditable well-pattern designs. Key innovations include: (i) a grouped geometric transformation scheme that enumerates baseline five-spot layouts with minimal redundancy; (ii) K-dimensional tree (KD-tree) spatial indexing for rapid and scalable matching of thousands of legacy wells to candidate vertices; and (iii) the explicit formalization of expert rules into a constrained multi-objective function with exact penalty methods. A standardized evaluation protocol is used to compare DE, PSO, and SLSQP across different spacing regimes, measuring solution quality, computational efficiency, and robustness over multiple random seeds. Results show that when applied to a large onshore oilfield, the framework reduced baseline plan generation from more than two months to less than 1 h, with each optimization scenario converging within minutes. DE exhibits superior robustness and pattern uniformity. SLSQP achieves the fastest convergence in smooth, lightly constrained landscapes but frequently stalls in local minima under complex regimes, leaving residual violations. PSO offers intermediate performance at a higher and more variable computational cost. The proposed framework serves as a field-deployable tool that substantially reduces the engineering design cycle, improves pattern quality, and enhances the reuse of existing infrastructure.

© 2026 The Authors. Publishing services by Elsevier B.V. on behalf of KeAi Communications Co. Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

1. Introduction

The sustainable development of mature oilfields remains critical to the global energy supply, as these fields hold a large share of the world's remaining hydrocarbon resources (IEA, 2023). After primary production, operators increasingly rely on secondary and tertiary recovery methods, including waterflooding and chemical flooding, to sustain production rates and increase ultimate recovery (Xu et al., 2020; Chappell, 2020). In this stage, infill drilling and well-pattern optimization are critical tools for improving sweep efficiency, accessing bypassed oil, and controlling water cut (Green and Willhite, 2018; Xu et al., 2007). However, designing an effective infill well pattern is not a simple geometric placement problem.

In mature reservoirs, well-pattern performance is strongly affected by reservoir anisotropy, directional permeability, natural or hydraulic fractures, and the relative orientation of producers and injectors (Liu, 2008; Guo et al., 2022). At the same time, redevelopment must account for dense legacy well infrastructure and operational rules accumulated over decades of field production. Practical constraints typically include type-dependent

* Corresponding author.
E-mail address: chenghaoran@qust.edu.cn (H.-R. Cheng).
Peer review under the responsibility of China University of Petroleum (Beijing).

https://doi.org/10.1016/j.petsci.2026.08.004
1995-8226/© 2026 The Authors. Publishing services by Elsevier B.V. on behalf of KeAi Communications Co. Ltd. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

Nomenclature

W_{old}	Set of legacy wells (eligible for reuse after filtering)
F	Set of fault polygons (volumetric/areal constraints)
Ω	Lease or sub-block boundary polygon (feasible area)
V	Set of pattern vertices (candidate well locations)
L_{JJ}	Set of injector-producer links (JJ segments)
C	Set of hard constraints (containment, spacing, faults, exclusions)
i, j	Vertex (or link) indices
t_i	Well type at vertex i (e.g., producer/injector)
$\mathbf{x}$	Decision vector (coordinates of all vertices)
S	Target pattern spacing, m
θ	Pattern angle relative to reference axis, deg
ϕ	Rotation applied to base grid, deg
$\Delta x, \Delta y$	Global translation of grid, m
$d_{ij}(x)$	Length of JJ connection between vertices i and j, m
λ_k	Objective weights in scalarization ($k = 1, 2, 3, 4$)
$\widehat{f}_k$	Objective f_k normalized by baseline statistics
ρ_{buffer}	Safety buffer distance to faults, m
$\rho\left(t_i, t_j\right)$	Minimum allowed separation between types t_i, t_j, m
∂F	Boundary of fault polygons
$\varepsilon_{\text{same}}$	Max distance for admissible same-type reuse, m
$\varepsilon_{\text{diff}}$	Max distance for admissible cross-type reuse (in m), typically $\leq \varepsilon_{\text{same}}$
$\text{reuse}_{\text{all}}$	Total number of legacy wells reused
$\text{reuse}_{\text{same}}$	Count of same-type reuses (producer→producer, injector→injector)

minimum spacing between wells, preferential reuse of existing wells, lease and geological boundaries, and avoidance of faults or exclusion zones. These requirements create a constrained and multi-objective design problem in which competing goals must be balanced, such as maintaining regular producer-injector spacing, maximizing same-type well reuse, minimizing deviation from idealized pattern geometry, and avoiding geological barriers. Current workflows handle this by parameterizing well trajectories and stimulation geometry (e.g., Hough-transform descriptors for hydraulically fractured horizontal wells) and then applying ensemble-based optimization. These workflows explore design alternatives across geological realizations while honoring operational constraints and production objectives (Xue et al., 2021).

The resulting optimization landscape is highly irregular. Continuous variables, including well coordinates, spacing, and pattern orientation, interact with discrete reuse decisions and non-smooth constraints such as fault intersections and proximity thresholds. Small changes in well position may alter reuse assignments, trigger spacing violations, or change whether a candidate pattern intersects a geological barrier. Consequently, manual design becomes inefficient and difficult to reproduce, especially in large mature fields containing thousands of wells and complex fault networks. In current practice, well patterns are often configured using geographic information system (GIS) and reservoir engineering tools and then revised through repeated multidisciplinary review cycles, which can take months and may depend heavily on individual engineering experience (Wang and Chen, 2019; Zhan et al., 2022; Zhao et al., 2022). These limitations motivate the development of automated, auditable, and rule-aware optimization methods for field-scale redevelopment planning.

Computational optimization for well placement and field development has been studied extensively. Early work focused on optimizing a small number of wells using simulation proxies, geometric indicators, or classical optimization algorithms (Bittencourt and Horne, 1997; Beckner and Song, 1995). However, the combinatorial number of patterns makes exhaustive dynamic simulation prohibitively expensive, so practitioners have adopted heuristic optimizers to guide the search and cut simulator evaluations to the low hundreds while preserving decision quality (Levchenko and Iskakov, 2023). Later studies introduced global and heuristic search methods, including genetic algorithms, particle swarm optimization (PSO), differential evolution (DE), swarm intelligence algorithms, and hybrid metaheuristics, often coupled with surrogate models to reduce the cost of full-physics reservoir simulation (Xu et al., 2007; Bouzarkouna et al., 2012; Chen et al., 2018; Kumar, 2021; Islam et al., 2020; Li et al., 2023). In mature fields, work has focused on decision making under uncertainty and reducing computational cost: Wang et al. (2012) introduced retrospective optimization for well-location planning, Wang and Chen (2019) proposed an integrated workflow for infill well placement in tight oil reservoirs consistent with multi-well pad fracture-placement co-design, and de Brito and Durlofsky (2020) developed a two-step surrogate treatment (ST) for well-control optimization. Empirical and numerical studies complement these algorithmic advances: Wang et al. (2022) systematically compared horizontal, staggered, and combined patterns under varying reservoir heterogeneity to guide offshore pattern selection, and Wu et al. (2025) reported application-specific insights from numerical simulation.

More recently, machine-learning-assisted optimization has become increasingly important with neural networks, XGBoost models, graph-based proxies, and deep-learning surrogates being used to accelerate well placement, well control, and EOR design under uncertainty (Eltayeb et al., 2025; Wang et al., 2023; Zhuang et al., 2025).

Despite these advances, important gaps remain for early-stage, field-scale redevelopment of mature assets. Many existing approaches optimize physics-based objectives such as net present value or cumulative oil production through repeated reservoir simulations. These methods are valuable for detailed evaluation and final decision refinement, but their computational cost limits the number of design alternatives and geological realizations that can be explored (Mao and Ghahfarokhi, 2024). Other studies simplify operational and geological constraints into basic well-distance rules, without fully representing polygonal barriers, lease boundaries, fault avoidance, type-dependent reuse policies, or large legacy well databases. Classical constrained multi-objective evolutionary algorithms, such as NSGA-II and MOEA/D, are effective for approximating Pareto fronts (Deb et al., 2002; Zhang and Li, 2007), but their population size, diversity maintenance, and long evolutionary runs may be impractical when rapid feasibility screening is the primary objective. Therefore, a complementary approach is needed: one that emphasizes scalable geometric evaluation, explicit rule enforcement, and robust optimization under realistic field constraints.

This study addresses this need by developing a geometry-driven and rule-aware optimization framework for infill well-pattern design in mature oilfields. Rather than replacing high-fidelity reservoir simulation, the proposed framework provides a scalable screening and optimization layer for early redevelopment planning. It formalizes engineering heuristics and operational

policies as quantifiable objectives and exact penalty constraints, enabling rapid generation of feasible and auditable well-pattern designs before subsequent simulation-based evaluation.

The main contributions of this work are as follows. First, a mathematical formulation is developed for constrained multi-objective well-pattern design, incorporating geometric regularity, same-type legacy well reuse, type-dependent spacing rules, and geological barrier avoidance. Second, an efficient computational strategy is introduced by combining grouped geometric transformations with KD-tree spatial indexing, allowing thousands of candidate patterns to be evaluated against large legacy well datasets. Third, DE, PSO, and SLSQP are systematically compared under multiple spacing regimes and repeated runs, providing practical evidence on solver robustness, feasibility, runtime, and tuning effort in constraint-rich field-scale applications. Finally, the proposed method is implemented as an end-to-end, configuration-driven workflow that produces reproducible and auditable outputs in standardized formats, supporting more efficient redevelopment planning for mature oilfields.

2. Methodology

Fig. 1 illustrates the overall workflow of the proposed rule-aware well-pattern optimization framework. The framework begins with field-scale spatial inputs, including legacy wells, fault polygons, and lease or sub-block boundaries, which are cleaned, filtered, and indexed to support efficient geometric queries. Candidate patterns are generated by enumerating grouped baselines and applying vectorized rotation and translation to predefined base grids. For each candidate, nearby legacy wells are identified using KD-tree-based spatial matching, after which rule-based reuse assignment is performed to prioritize same-type well reuse while satisfying distance constraints. The resulting candidates are then evaluated through a constrained multi-objective formulation that accounts for geometric regularity, fault avoidance, and legacy-well reuse. The scalarized objective is optimized using DE, PSO, and SLSQP, producing reproducible and auditable well-pattern designs together with performance metrics for subsequent engineering review or simulation-based assessment.

2.1. Field data, geometry, and preprocessing

The framework was applied to a large onshore mature oilfield with extensive legacy infrastructure. The dataset contained 1565 legacy wells, including 878 producers and 687 injectors, of which 40 wells were scrapped and excluded from reuse. The field comprised 10 reservoir layers and 78 interpreted faults, with faulting present in 9 layers.

The main inputs included legacy well data, fault polygons, and lease or sub-block boundaries. Legacy well (W_{old}) records contained well identifiers, coordinates, well types, and operational statuses. Scrapped or abandoned wells were removed to ensure that only operationally valid wells were considered for reuse. Fault polygons (F), derived from geological interpretation, represented forbidden subsurface zones and were used to check potential well path intersections. Lease or sub-block boundaries (Ω) defined the allowable area for new pattern vertices. When available, stratigraphic eligibility criteria, such as target reservoir-units, were also used to retain only wells completed in relevant zones.

All input data were cleaned, standardized, and spatially indexed before optimization. Preprocessing included unifying well-type and status labels, resolving coordinate inconsistencies, and computing bounding boxes for lease and fault polygons. These indexes enabled rapid exclusion of infeasible candidate vertices before more expensive point-in-polygon or line-intersection tests. For fields divided into sub-blocks, legacy wells were assigned to sub-blocks using point-in-polygon spatial joins, allowing

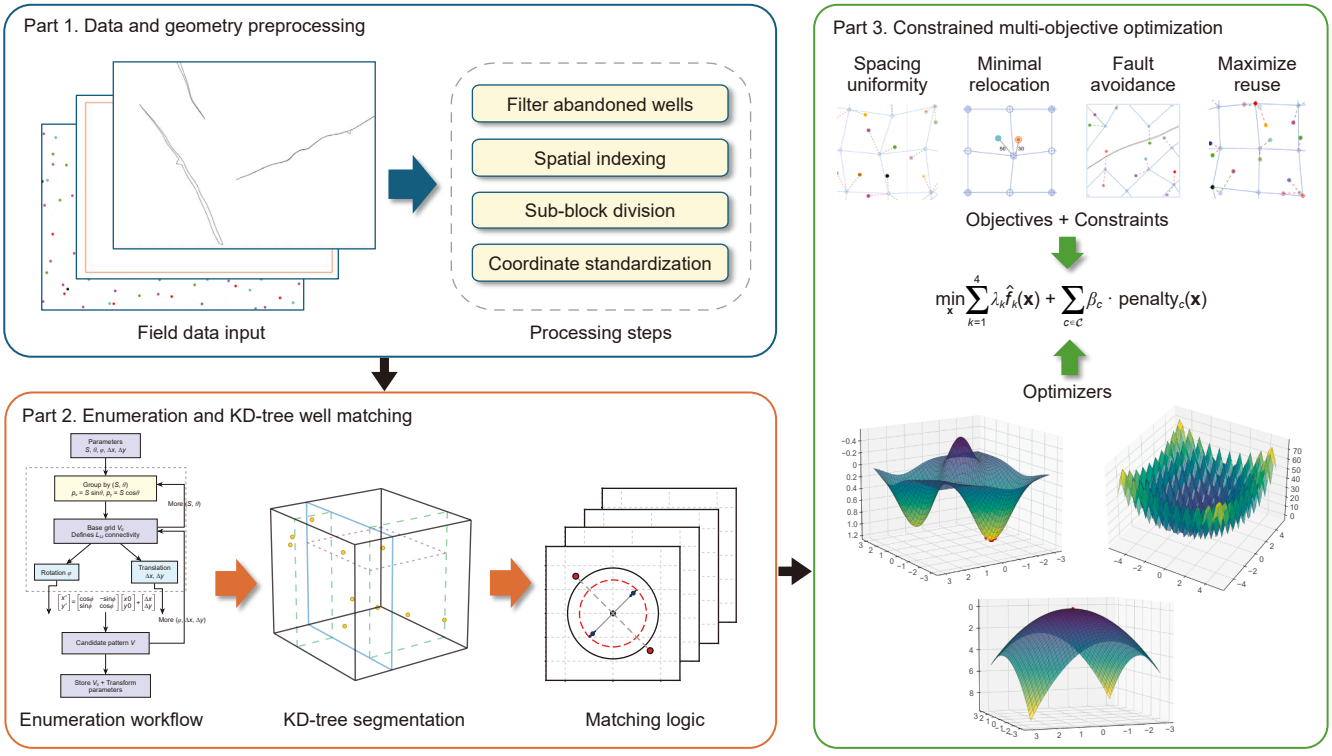


Fig. 1. Integrated framework architecture for intelligent well-pattern optimization.

optimization to be performed separately within each compartment and reducing computational complexity.

2.2. Grouped baseline enumeration

A direct approach for for candidate-pattern generation would enumerate a complete grid for every combination of spacing (S), pattern angle (θ), rotation (ϕ) and translation (Δx, Δy). However, this approach was deemed computationally expensive for fine parameter sampling. To reduce redundant calculations, this study adopts a grouped baseline enumeration strategy.

The full Cartesian product of parameters was denoted as E. Parameters that determine the intrinsic grid structure, such as spacing and pattern pitch, were grouped together as (S, θ, p_x, p_y), where $p_x = S \sin\theta$ and $p_y = S \cos\theta$, represented the pitches in the x and y directions. For each unique group (S,θ), a single base grid V_0 was generated and centered at the origin (0, 0). This base grid defined the fundamental connectivity of the pattern, specifically the JJ links (L_{JJ}) for the five-spot pattern.

For each base grid V_0, different rotations (ϕ) and translations (Δx, Δy) were applied to generate candidate layouts. The vertex coordinates for each candidate pattern were calculated using a standard rotation-translation transformation, implemented as a vectorized matrix operation over the full coordinate array:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \quad (1)$$

As illustrated in Fig. 2, this grouped strategy ensures that the computationally expensive construction of grid connectivity is performed only once for each (S,θ) group, rather than repeated for every $(S, \theta, \phi, \Delta x, \Delta y)$ combination. Consequently, the method substantially reduces both computation time and memory usage, since only the base grids and corresponding transformation parameters need to be stored.

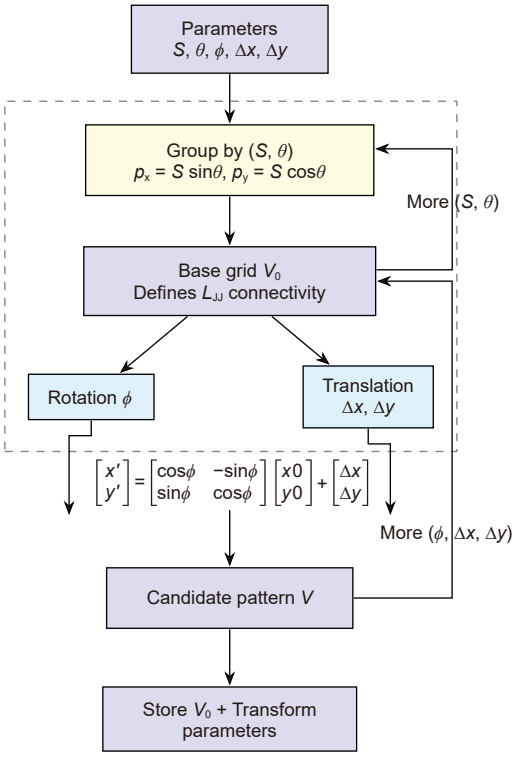


Fig. 2. Schematic of the grouped enumeration workflow.

2.3. KD-tree matching and reuse logic

For each rotated and translated candidate pattern V, the reuse of existing legacy wells W_{old} was evaluated. A brute-force approach calculating distances from every legacy well to every vertex would result in $O(|W_{old}| \cdot |V|)$ complexity, which would become impractical for large field-scale cases where $|W_{old}|$ could exceed 5000 and $|V|$ could exceed 1000. To improve efficiency, a KD-tree was constructed over the candidate vertices. Building the KD-tree requires $O(|V|\log|V|)$ time, and each nearest-neighbor query has an average complexity of $O(\log|V|)$. The overall matching cost is therefore reduced to approximately $O(|V|\log|V|+|W_{old}|\log|V|)$, making the procedure suitable for large-scale applications.

For each legacy well $w_i \in W_{old}$, the KD-tree was queried for its four nearest candidate vertices (Fig. 3). Each candidate match (w_i, v_j) was then tested against distance and type-compatibility rules. If the legacy well and candidate vertex had the same intended type, the Euclidean distance $r(w_i, v_j)$ was required to be below the same-type threshold ε_{same}. If cross-type reuse was permitted, the distance had to satisfy the stricter threshold ε_{diff}, where $\varepsilon_{diff} \le \varepsilon_{same}$, reflecting the higher operational risk and cost of wellbore conversion.

Querying four neighbors, rather than only the closest vertex, provides alternative feasible matches for the subsequent assignment step. All admissible matches were sorted by priority and distance, with same-type matches ranked ahead of cross-type matches and shorter distances preferred within each priority class. A deterministic greedy one-to-one assignment was then performed: the highest-priority admissible match was selected, the corresponding candidate vertex was marked as used, and all remaining matches involving that vertex were discarded. Although this procedure does not guarantee a globally optimal assignment, it ensures a feasible and reproducible reuse configuration that favors close, same-type matches.

The output of this process consisted of two key counters: $reuse_{all}$, which measured the total number of legacy wells assigned to any pattern vertex, $reuse_{same}$, which counted the number of legacy wells assigned to a vertex of the exact same type.

2.4. Objectives and constraints

The well-pattern design problem was formulated as a constrained multi-objective optimization problem and then scalarized into a single composite objective using weighted sums and exact penalties, a widely used strategy in engineering design optimization (Nocedal and Wright, 2006). The decision variable is the vector **x** containing the coordinates of all pattern vertices. Scalarization enables the use of efficient single-objective solvers while retaining multiple design criteria related to geometry, operational feasibility, and legacy-well reuse. Four objectives were considered.

O1: Spacing uniformity (f_1).

An effective flood pattern requires relatively uniform injector-producer spacing to maintain stable flood fronts and reduce the risk of early water breakthrough. Therefore, the standard deviation (SD) and coefficient of variation (CV) of the lengths of all J-J connections were calculated to ensure the even distribution of planned wells. In practice, SD is used as the primary objective, while CV is reported as a scale-normalized diagnostic metric.

O2: Minimal relocation (f_2).

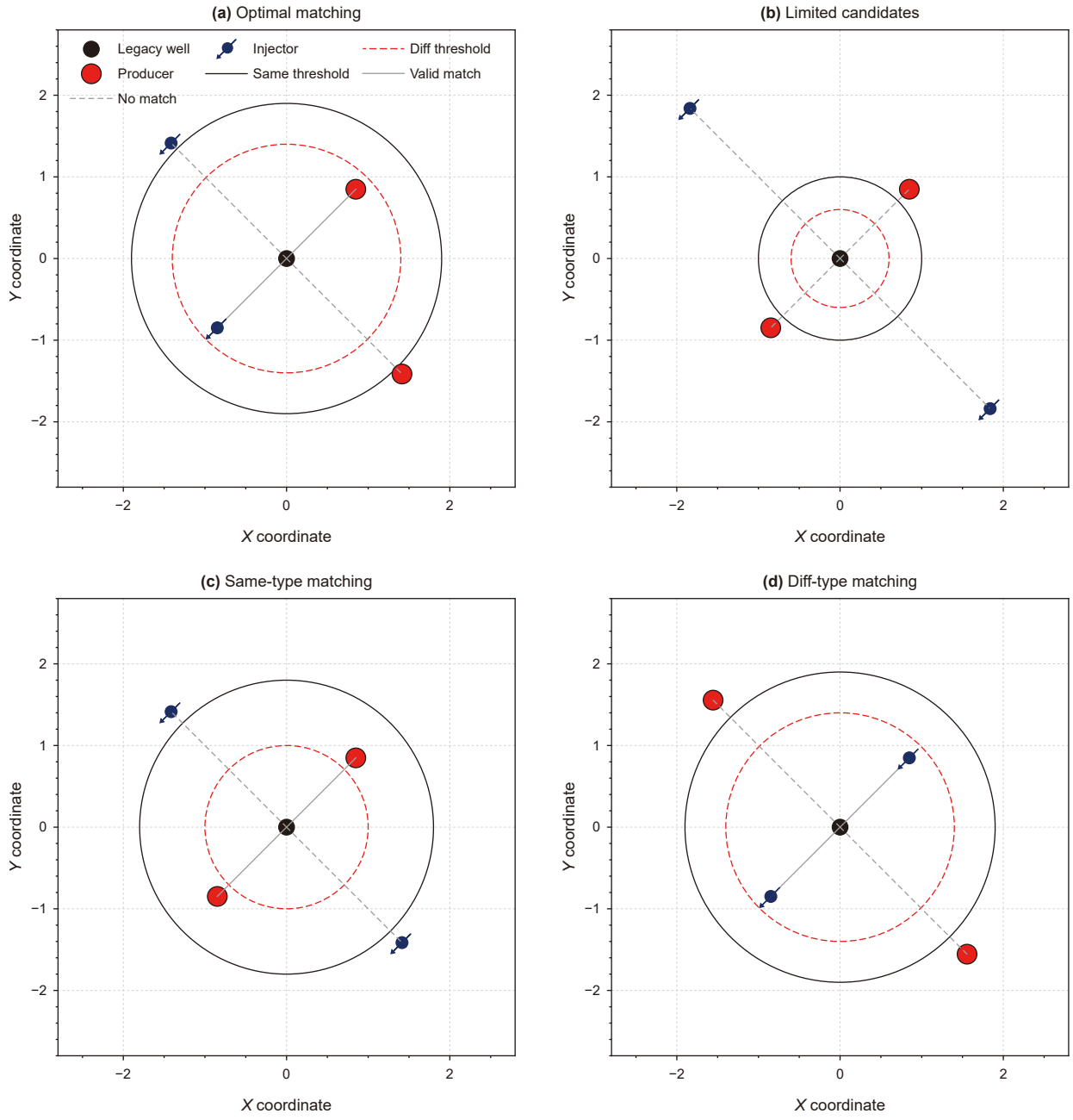


Fig. 3. KD-tree spatial matching scenarios.

Deviation of a vertex from its initial enumerated position implies non-ideal reuse and may incur additional costs associated with site preparation, flowline rerouting, and environmental impact. Therefore, the total displacement of all vertices should be minimized.

O3: Fault avoidance (f_3).

Pattern segments that intersect fault polygons or lie within a specified buffer distance from faults introduce elevated geomechanical and operational risks. Such intersections and near-fault configurations are penalized to enforce a safe offset from fault boundaries.

O4: Legacy well reuse (f_4).

Reuse of existing wells was encouraged to reduce capital expenditure and operational risk. Same-type reuse (producer-to-producer or injector-to-injector) is weighted more heavily than cross-type reuse because its lower technical and economic uncertainty.

$$\begin{cases} f_1(\mathbf{x}) = \sigma_d\left(d_{ij}(\mathbf{x})\right) \\ f_2(\mathbf{x}) = \sum_{i\in V} \parallel \mathbf{x}_i - \mathbf{x}_{i_{\text{init}}} \parallel_2 \\ f_3(\mathbf{x}) = \sum_{l\in L_{\text{JJ}}} \left[\mathrm{I}_{l\cap \mathrm{F}\neq\varnothing} + \alpha\cdot\max\left(0, \rho_{\text{buffer}} - \text{dist}(l,\partial\mathrm{F})\right)\right] \\ f_4(\mathbf{x}) = -\left[w_{\text{same}}\cdot\text{reuse}_{\text{same}} + w_{\text{diff}}\cdot(\text{reuse}_{\text{all}} - \text{reuse}_{\text{same}})\right] \end{cases} \tag{2}$$

where $\mathbf{x}_{i_{\text{init}}}$ represents the coordinates of vertex i in the initial enumeration mode, I is an indicator function (1 if true, 0 otherwise), α is a scaling factor, ρ_{buffer} is the desired safety distance from faults, and $w_{\text{same}} > w_{\text{diff}}$ are positive weights. Hard constraints were implemented using exact penalty methods. These included the requirement that all vertices must lie within the lease or sub-block boundary, i.e., $(\mathbf{x}_v, \mathbf{y}_v) \in \Omega \; \forall \, v \in V$. A minimum separation distance between any pair of vertices $\left(v_i, v_j\right)$ with types $\left(t_i, t_j\right)$ was enforced to prevent well interference, i.e., $|\mathbf{x}_{v_i} - \mathbf{x}_{v_j}|_2 \geq \rho\left(t_i, t_j\right)$. Additionally, vertices were prohibited from being placed inside excluded polygons, such as existing surface pads, facilities, or environmentally protected zones. The scalarized objective function was formulated as:

$$\min_{\mathbf{x}} \sum_{k=1}^{4} \lambda_k \widehat{f}_k(\mathbf{x}) + \sum_{c \in C} \beta_c \cdot \text{penalty}_c(\mathbf{x}) \tag{3}$$

where $\widehat{f}_k$ represented the objectives normalized by their typical values in the initial baseline population to ensure commensurability. λ_k were user-defined weights reflecting the relative importance of each objective. In practice, larger weights are assigned to spacing uniformity and fault avoidance during early-stage pattern screening, while legacy-well reuse is given higher weight in redevelopment scenarios emphasizing cost reduction, with relocation typically assigned a moderate weight. β_c were sufficiently large constant penalty coefficients enforcing strict satisfaction of the hard constraints $c \in C$ (e.g., β_{fault} for fault intersections, β_{contain} for boundary violations). Here, $\text{penalty}_c(\mathbf{x})$ represents the exact penalty function that quantifies the magnitude of violation of constraint c. For inequality constraints of the form $g_c(\mathbf{x}) \leq 0$, the penalty is defined as $\text{penalty}_c(\mathbf{x}) = \max\{0, g_c(\mathbf{x})\}$, while equality constraints $h_c(\mathbf{x}) = 0$ are penalized as $\text{penalty}_c(\mathbf{x}) = |h_c(\mathbf{x})|$. In this study, all hard constraints are formulated as inequality constraints and enforced through exact penalties. The values of β_c were set larger than the maximum possible improvement in the objective function to ensure constraint satisfaction was always prioritized (Nocedal and Wright, 2006).

A weighted-sum formulation was adopted for computational efficiency and interpretability in large-scale, constraint-rich screening problems. Although different weight combinations may identify only a subset of Pareto-optimal solutions, the objective of this framework is not to reconstruct the entire Pareto front. Instead, it aims to identify practically relevant, feasible, and engineering-preferred well-pattern configurations under explicit operational priorities. In field applications, these priorities are usually defined by development strategy and regulatory requirements, which substantially reduces the need for exhaustive trade-off exploration. The combination of objective normalization and feasibility-first exact penalties enables representative weight settings to produce stable optimization, allowing the exploration of alternative priority scenarios without the prohibitive computational cost associated with ϵ-constraint-based methods.

2.5. Modified optimizers

Three optimization algorithms were systematically implemented with enhancements to address the complex, discontinuous, and high-dimensional nature of well-pattern optimization. Each algorithm was specifically tailored to overcome the unique challenges posed by the mixed-integer, non-convex optimization landscape with exact penalty constraints.

2.5.1. Differential evolution with adaptive constraint handling

Differential evolution (DE) is a gradient-free global optimization algorithm and is well suited to problems involving discontinuities caused by exact penalty terms and discrete assignment changes (Wang et al., 2023; Chen et al., 2018). In this study, an improved DE variant based on the classical "DE/rand/1/bin" strategy was developed with adaptive parameter control and a constraint-handling mechanism. The mutation step was enhanced through a self-adaptive scaling factor:

$$F_i^{t+1} = F_{\text{l}} + (F_{\text{u}} - F_{\text{l}}) \cdot U(0,1) \cdot \exp\left(-\gamma \cdot \frac{\text{feas}_i^t - \text{feas}_{\text{best}}^t}{\text{feas}_{\text{best}}^t} \right) \tag{4}$$

where F_{l} and F_{u} define the lower and upper bounds of the scaling factor, feas_i^t represents the constraint violation measure of individual i at generation t, and γ controls the adaptation rate. This formulation automatically adjusts the exploration intensity based on constraint satisfaction progress.

2.5.2. Particle swarm optimization with multi-swarm topology

Particle swarm optimization (PSO) is a conceptually simple population-based optimizer that often exhibits rapid early-stage convergence, although it may suffer from premature convergence in complex or highly multimodal search spaces (Islam et al., 2020; Kumar, 2021). In this study, a dynamic multi-swarm PSO variant with quantum-inspired tunneling was implemented. The velocity update incorporated both personal-best and neighborhood-best information:

$$v_i^{t+1} = \omega(t) v_i^t + c_1 r_1 \left(p_i - x_i^t\right) + c_2 r_2 \left(l_i - x_i^t\right) + c_3 r_3 \left(g - x_i^t\right) + \xi \cdot Q(t) \tag{5}$$

where $\omega(t)$ provides linear inertia decay, l_i represents the local best from a dynamically constructed Von Neumann neighborhood, and $Q(t)$ introduces quantum tunneling probability to escape local minima:

$$Q(t) = \sqrt{\frac{-2\ln(u_1)}{m}} \cdot \cos(2\pi u_2) \cdot \exp(-\eta \cdot \text{diversity}(t)) \tag{6}$$

where $u_1, u_2 \sim U(0, 1)$, m is the problem dimension, and $\text{diversity}(t)$ measures population spread. The swarm periodically undergoes topological reorganization based on fitness-distance correlation analysis.

Constraint handling in the PSO implementation followed a feasibility-first, penalty-based strategy consistent with the exact-penalty formulation in Section 2.4. At each iteration, particle positions are updated using the standard velocity–position equations, and all candidate solutions are evaluated with the scalarized objective function in Eq. (3), which includes exact penalty terms for hard constraint violations. Feasible particles were preferred over infeasible ones during personal-best and global-best solutions. Among infeasible particles, solutions with smaller total constraint violations were preferred. As a result, particles violating geometric or operational constraints incur large penalty values and are systematically disfavored, allowing the swarm to explore the search space while progressively converging toward the feasible region without requiring explicit projection or repair operators.

2.5.3. SLSQP with surrogate gradient enhancement

Sequential least squares programming (SLSQP) is a gradient-based method that solves successive quadratic programming subproblems using quasi-Newton Hessian approximations and is effective for smooth problems with moderate constraint sets (Nocedal and Wright, 2006). In this study, SLSQP was implemented

using the SciPy's minimize function. To improve robustness under non-smooth geometric constraints and exact penalties, a hybrid SLSQP framework is employed through problem-level enhancements rather than modifying the solver kernel. Specifically, non-differentiable indicators in fault-intersection and buffer-distance constraints were replaced multi-scale smooth surrogate functions:

$$\mathrm{I}_{l\cap \mathrm{F}\neq\varnothing} \approx \frac{1}{2}+\frac{1}{\pi}\arctan\Big(k(t)\cdot\big(\rho_{\mathrm{buffer}}-\mathrm{dist}(l,\partial\mathrm{F})\big)\Big) \tag{7}$$

where $k(t)$ provides adaptive smoothing intensity. Small values of $k(t)$ are used in early iterations to ensure smooth gradients, and $k(t)$ is gradually increased as optimization progresses to better approximate the original non-smooth constraints.

Gradients were evaluated at the problem level. Analytical gradients were used for smooth objective terms, such as spacing uniformity and vertex displacement. For originally non-differentiable geometric constraints, adaptive smooth surrogates were introduced, and their gradients were derived analytically. When closed-form gradients were unavailable, numerical differentiation was applied consistently.

The Quasi-Newton Hessian approximation required by SLSQP was handled internally by the solver, without explicit Hessian construction or modification. Constraint handling followed a feasibility-first exact-penalty strategy: early iterations were dominated by penalty terms to promote feasibility, whereas later iterations emphasized objective improvement once major violations had been reduced. This allowed SLSQP to handle constraint-rich and partially non-smooth landscapes without explicit projection or repair operators.

2.6. Scenarios and metrics

To evaluate algorithm performance under different levels of constraint intensity, three spacing scenarios were defined to represent progressive stages of infill development. Scenario A (target spacing $S = 150$ m) used a large spacing, producing fewer pattern vertices and a sparser constraint landscape, and represents early development or reservoirs with favorable flow properties. Scenario B (target spacing $S = 124$ m) used an intermediate spacing, corresponding to a typical first infill campaign with increased well density and stronger constraint interactions. Scenario C (target spacing $S = 90$ m) used a tight spacing, representing aggressive infill development or tertiary recovery, where dense patterns make minimum-spacing and fault-avoidance constraints difficult to satisfy.

Fig. 4 shows the convergence behavior of DE, PSO, and SLSQP under the three scenarios. The algorithms performed similarly in the low-difficulty case, whereas DE showed more robust convergence as constraint interactions intensified. In contrast, SLSQP was more likely to stall under tightly constrained conditions.

For each optimized solution, the following metrics were computed (Table 1), selected for their practical relevance to field application. Pattern uniformity was measured by the standard deviation (SD) and the coefficient of variation (CV) of the distances between injectors and producers (J-J distances), with lower values indicating better regularity. Constraint satisfaction was quantified by the number of unoptimized points, defined as residual violations of minimum-separation constraints after optimization, with a target value of zero. Computational efficiency was measured as the total runtime required to meet the stopping criteria.

A consistent evaluation protocol was applied to all scenarios. First, grouped baseline enumeration was performed over predefined parameter ranges, including spacing variation pattern angle, rotation step, and translation steps. Each baseline pattern was evaluated using the scalarized objective function without vertex relocation. The top 20 baseline configurations were then selected as initial candidates for optimization. DE and PSO were each run 20 times with different random seeds, using populations initialized around the best baseline pattern. SLSQP was run 20 times, each initialized from one of the top 20 baseline configurations.

To support cross-scenario comparison, an overall performance score was calculated from normalized runtime, uniformity, and constraint-satisfaction metrics. Each metric was scaled to 0-100 within each scenario using reverse scaling, so that higher values indicated better performance. The final score used engineering-priority weights of 40% for constraint satisfaction, 35% for pattern uniformity, and 25% for runtime. Results were

Table 1
Formal definitions of evaluation metrics.

Metric	Definition
Uniformity (SD)	Std. dev. of all J-J distances
Uniformity (CV)	Coefficient of variation of J-J distances
Unoptimized points	Count of min. separation violations
Runtime	Wall-clock time to convergence
Rule compliance	% segments passing all constraints

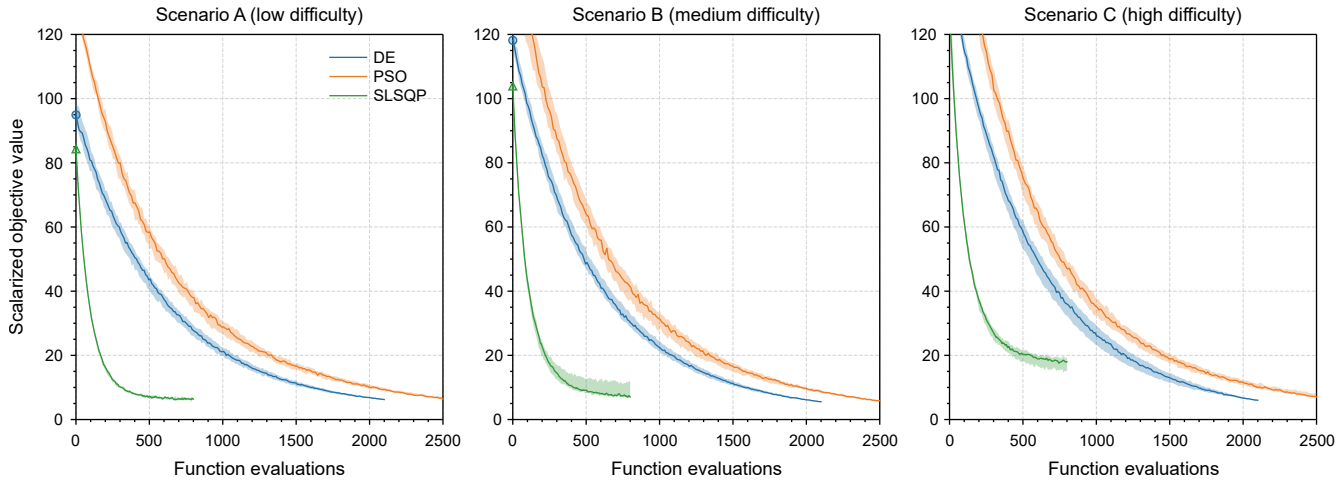


Fig. 4. Schematic diagrams of the convergence curves for each algorithm and scenario.

aggregated across random seeds for DE and PSO and across initial points for SLSQP. Mean, median, standard deviation, and interquartile range (IQR) were reported to summarize both performance and robustness.

3. Results and discussion

This section presents the performance of the proposed rule-aware well-pattern optimization framework in a real-field case. Three algorithms (DE, PSO, and SLSQP) were systematically compared under different spacing scenarios to evaluate their convergence characteristics, robustness, computational efficiency, and ability to satisfy complex geometric and operational constraints.

3.1. Visual analysis of optimized patterns

Fig. 5 compares the optimized patterns generated by the three algorithms. Black lines represent faults, blue lines indicate injector-producer connections, and colored points denote legacy wells of different types. Since the candidate patterns were generated from rotated and translated base grids, the optimized layouts were not restricted to a fixed orthogonal orientation.

In the most constrained case, Scenario C, DE produced regular and continuous J-J connections while avoiding fault buffers with limited geometric distortion. This indicates that DE was able to adjust the pattern around local constraints while maintaining overall structural regularity. PSO generated visually comparable patterns in several runs, but some solutions required greater local distortion and occasionally retained isolated violations. In contrast, SLSQP often produced localized irregularities near faults or neighboring vertices, suggesting convergence to suboptimal local configurations close to the initial baseline. These visual patterns are consistent with the quantitative results and highlight the greater robustness of DE in tightly constrained search spaces.

3.2. Productivity and rule-checking accuracy

The automated framework substantially improved engineering productivity (Table 2). More than 50,000 baseline patterns were generated and screened in less than 45 min, whereas comparable manual workflows historically required approximately two months for a field of this scale. Subsequent optimization for each scenario converged within seconds to minutes, reducing the analysis time by more than three orders of magnitude.

The framework also achieved high rule-checking accuracy. The error rate was defined as the fraction of design elements that violated predefined rules or required correction during review, including spacing violations, fault-buffer infringements, and incorrect legacy-well reuse assignments. Manual error rates were

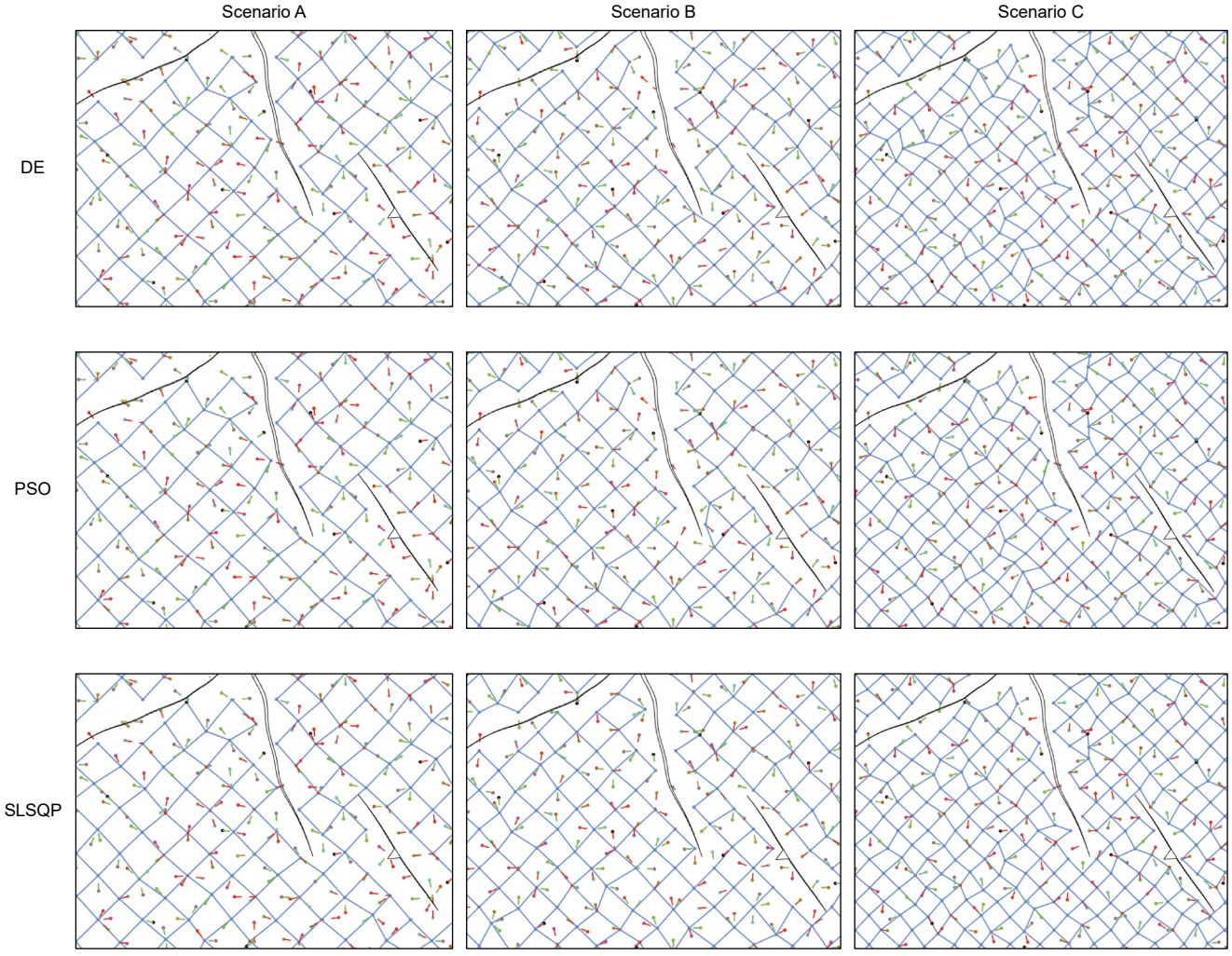


Fig. 5. Optimized patterns from DE, PSO, and SLSQP.

Table 2
Comparison of key productivity indicators.

Metric	Manual process	Automated framework	Improvement
Time per scenario	~60 days	~45 min	2000x faster
Design cycles	3–4 iterations	1000s of patterns evaluated	300x more options
Error rate	15%–20% (est.)	0%–1.6%	10x more accurate
Reuse identification	Subjective	Quantitative, consistent	100% reproducible
Constraint checking	Incomplete	Exhaustive, automated	-

estimated from historical review records, whereas automated error rates were calculated from residual constraint violations in optimized outputs. Under this definition, the automated rule-checking module achieved an accuracy above 98.4%, with most remaining discrepancies associated with edge cases near fault traces. This result confirms that the proposed framework can translate expert design rules into reproducible and auditable computational checks.

3.3. Comparative analysis by spacing regime

Table 3 summarizes algorithm performance under the three spacing scenarios. In Scenario A, with 150 m spacing, all algorithms performed well. SLSQP was the fastest at 5.56 s, followed by DE at 21.45 s and PSO at 33.30 s. All methods achieved 0–1 unoptimized points and no fault intersections, with nearly identical pattern uniformity. The SD values were 6.20 m for DE, 6.19 m for SLSQP, and 6.25 m for PSO. These results indicate that, in relatively simple settings where the constraint landscape is smooth and the baseline pattern is already close to feasible, a well-initialized local optimizer such as SLSQP can provide sufficient solution quality with the lowest runtime.

In Scenario B, SLSQP remained the fastest at 15.13 s, while DE and PSO required 44.36 s and 73.10 s, respectively. However, robustness differences became more evident. SLSQP stalled in local minima in 4 out of 20 runs, producing an average of 4 residual unoptimized points. Both DE and PSO consistently found feasible solutions with zero unoptimized points. DE also achieved slightly better uniformity than PSO, with an SD of 5.52 m and CV of 4.44%, compared with 5.57 m and 4.48% for PSO.

In Scenario C, tighter spacing increased runtime for all solvers and clearly separated their constraint-handling capability. SLSQP remained the fastest at 54.72 s, followed by DE at 112.46 s and PSO at 178.77 s. However, SLSQP became trapped in poor local minima, producing an average of 21 unoptimized points. PSO improved upon SLSQP but still retained an average of 3 unoptimized points. DE was the only algorithm that consistently produced fully feasible solutions, with zero unoptimized points, zero fault violations, and the best uniformity, achieving an SD of 5.95 m and a CV of 7.40%.

Fig. 6 further summarizes the comparative results across all scenarios. The weighted performance score integrates constraint satisfaction, pattern uniformity, and runtime using engineering-priority weights. DE achieved the highest overall score in every scenario, while SLSQP showed a marked decline as spacing constraints became more difficult. This confirms that DE provided the best overall adaptability across spacing regimes.

3.4. Algorithm stability and mechanistic interpretation

Fig. 7 compares algorithm stability across 20 initial conditions, including different random seeds for DE and PSO and different starting points for SLSQP. DE showed the most consistent performance, with narrow interquartile ranges for the scalarized objective, constraint penalty, and uniformity standard deviation. This indicates reliable convergence across different initial populations.

DE's superior robustness is primarily attributed to its population-based search mechanism. By maintaining multiple candidate solutions throughout optimization, DE can explore several promising regions simultaneously and reduce the risk of premature convergence. Its differential mutation operation, $F\cdot(X_{r2}-X_{r3})$, provides an implicit adaptation of step size: larger population differences promote global exploration in early iterations, while smaller differences support local refinement near feasible optima. This behavior is well suited to the non-smooth objective landscape created by exact penalties, discrete legacy-well reuse assignments, and fault-buffer constraints.

PSO exhibited intermediate behavior. It frequently found good solutions, but its performance was more sensitive to swarm initialization and parameter settings, particularly the inertia schedule. In penalty-dominated and partially discontinuous landscapes, early leader particles located near constraint boundaries may cause the swarm to aggregate prematurely or oscillate near infeasible regions. As a result, PSO showed higher run-to-run variability and longer runtimes than DE in tightly constrained scenarios, although it remains a viable option for smoother problems when carefully tuned.

SLSQP showed the highest effective variability. Although deterministic for a fixed starting point, its performance depended strongly on the quality of the initial baseline. In Scenario A, where the landscape was relatively smooth and the initial pattern was close to feasible, SLSQP converged rapidly and achieved competitive results. However, in Scenarios B and C, it often became trapped

Table 3
Comprehensive panel of per-pattern results for all scenarios and algorithms.

Scenario	Algorithm	Runtime, s	Uniformity SD, m	Uniformity CV, %	Unoptimized points
A	DE	21.45 ± 3.21	6.20	4.13	0
	PSO	33.32 ± 4.44	6.25	4.17	0
	SLSQP	5.56 ± 0.87	6.19	4.12	1
B	DE	44.36 ± 5.52	5.52	4.44	0
	PSO	73.10 ± 8.45	5.57	4.48	0
	SLSQP	15.13 ± 2.36	5.61	4.52	4
C	DE	112.46 ± 15.32	5.95	7.40	0
	PSO	178.77 ± 24.63	6.12	7.65	3
	SLSQP	54.72 ± 8.91	7.83	9.79	21

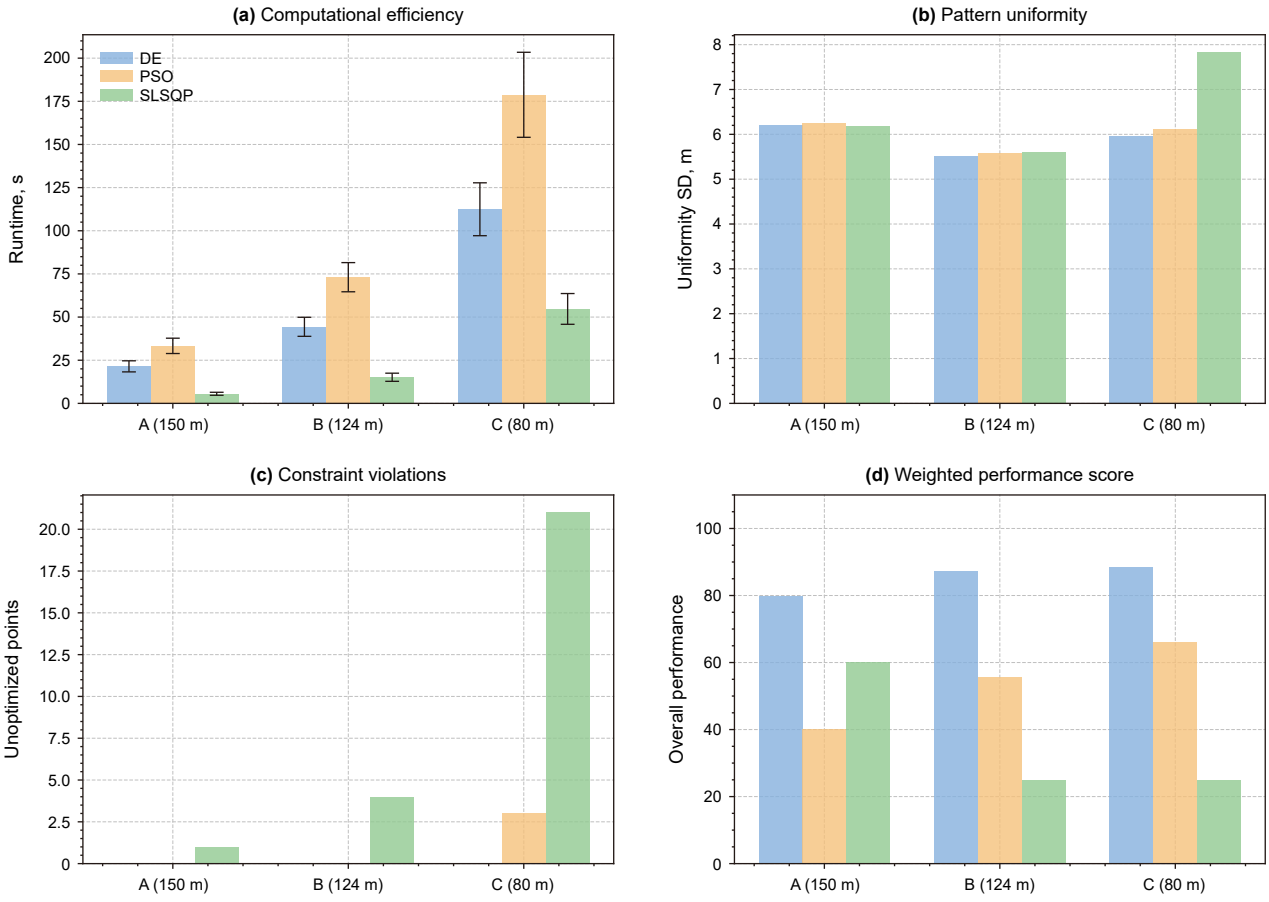


Fig. 6. Results and overall performance comparison.

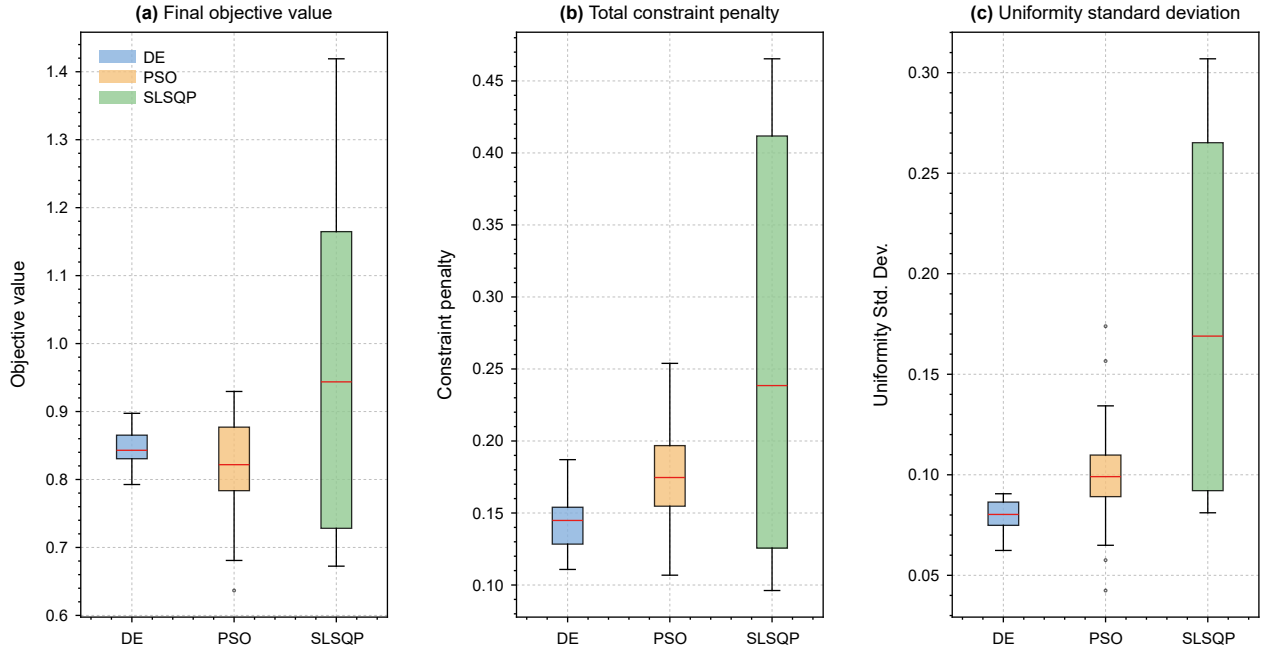


Fig. 7. Algorithm performance across different initial conditions.

in local minima and could not recover from unfavorable initializations. This suggests that SLSQP is useful for fast local refinement or simple field areas, but it is less reliable as a standalone optimizer for constraint-rich field-scale redevelopment.

3.5. Practical implications and limitations

The comparative results suggest a practical algorithm-selection strategy for field development applications. For most real-world infill projects involving tight spacing, dense fault networks, and strict operational constraints, DE is the most reliable default choice. Its modest computational cost, on the order of minutes, is small relative to the engineering time saved and is justified by its ability to consistently produce feasible, high-quality patterns. SLSQP is suitable for rapid screening in relatively simple areas or as a local refinement step when initialized from a high-quality baseline. A hybrid strategy, in which DE first identifies a promising region and SLSQP subsequently refines the solution, may offer a practical balance between global robustness and fast local convergence. PSO can serve as an alternative when prior experience with parameter tuning is available, but its higher variability makes it less reliable for production-scale deployment under tight constraints.

Despite these advantages, the proposed framework has several limitations. The current objectives are primarily geometric and operational, so final validation should be coupled with reservoir flow simulation to directly assess recovery performance and net present value. Future work will integrate fast physics-based surrogates, such as streamline models or deep-learning proxies, into the optimization loop. Additional surface and infrastructure constraints, including pipelines, pads, access roads, and terrain restrictions, could also be incorporated to improve practical applicability. Finally, this study considers a single development stage; extending the framework to multi-stage optimization of drilling sequence and pattern adjustment would support long-term field redevelopment planning.

4. Conclusions

This study presents a rule-aware and computationally efficient framework for automated well-pattern optimization in mature oilfields. Its main contributions include formalizing expert engineering rules within a constrained optimization model, using grouped geometric transformations and KD-tree spatial indexing for field-scale scalability, and systematically comparing optimization algorithms under realistic operational constraints.

Numerical experiments show that DE provides the most robust performance for this class of constraint-rich problems, consistently generating feasible, high-quality solutions in tight-spacing and fault-complex settings. SLSQP is effective for rapid drafting in simpler cases, and PSO remains viable with appropriate tuning; however, DE is the preferred option when feasibility and robustness are critical. Operationally, the framework reduces well-pattern design time from months to hours and produces reproducible, auditable outputs that can be integrated into existing engineering workflows.

More broadly, this work contributes to the digital transformation of upstream petroleum engineering by enabling systematic and reproducible well-pattern design. Future work will integrate physics-based surrogate models to include flow and economic objectives, incorporate geological uncertainty through multi-realization optimization, and extend the framework to jointly optimize well placement and control over the full asset lifecycle.

CRediT authorship contribution statement

Wan-Ming Xiong: Writing – original draft, Visualization, Methodology. **Bei-Bei Xing:** Validation, Resources. **Lian-Bin Zhong:** Validation, Resources. **Guang Wang:** Validation. **Xiao-Long Chen:** Investigation, Formal analysis. **Long Chen:** Validation, Data curation. **Rui Zhang:** Software. **Yu-Jie Tian:** Validation, Methodology, Conceptualization. **Rui Huang:** Resources, Investigation. **Hao-Ran Cheng:** Writing – review & editing, Supervision, Project administration, Funding acquisition, 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.

Acknowledgements

We gratefully acknowledge the financial supports from the National Science and Technology Major Project (No. 2025ZD1407001), the CNPC Innovation Fund (2024DQ02-0118), the Taishan Scholars Program of Shandong Province, China (No. tsqn202312208).

References

Beckner, B.L., Song, X., 1995. Field development planning using simulated annealing-optimal economic well scheduling and placement. In: SPE Annual Technical Conference and Exhibition 1995, p. SPE-30650. https://doi.org/10.2523/30650-MS.

Bittencourt, A.C., Horne, R.N., 1997. Reservoir development and design optimization. In: SPE Annual Technical Conference and Exhibition 1997, p. SPE-38895. https://doi.org/10.2118/38895-MS.

Bouzarkouna, Z., Ding, D.-Y., Auger, A., 2012. Well placement optimization with the covariance matrix adaptation evolution strategy and meta-models. Comput. Geosci. 16 (1), 75–92. https://doi.org/10.1007/s10596-011-9254-2.

Chappell, D., 2020. Waterflooding: Design and Development. Society of Petroleum Engineers, Richardson.

Chen, H., Feng, Q., Zhang, X., et al., 2018. Well placement optimization for offshore oilfield based on Theil index and differential evolution algorithm. J. Pet. Explor. Prod. Technol. 8 (4), 1225–1233. https://doi.org/10.1007/s13202-017-0403-6.

de Brito, D.U., Durlofsky, L.J., 2020. Well control optimization using a two-step surrogate treatment. J. Petrol. Sci. Eng. 187, 106565. https://doi.org/10.1016/j.petrol.2019.106565.

Deb, K., Pratap, A., Agarwal, S., et al., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6 (2), 182–197. https://doi.org/10.1109/4235.996017.

Eltayeb, A., Massarweh, O., Abushaikha, A., 2025. Machine learning in well placement optimization: A comprehensive review. Innov. Technol. Reserv. Optim. 2025 (1), 1–5. https://doi.org/10.3997/2214-4609.2025638051.

Green, D.W., Willhite, G.P., 2018. Enhanced Oil Recovery. Society of Petroleum Engineers, Richardson.

Guo, C., Fu, G., Liu, J., et al., 2022. Study on fracture development characteristics of low permeability oilfield and its influence on horizontal well pattern deployment. Chem. Technol. Fuels Oils 58 (2), 360–364. https://doi.org/10.1007/s10553-022-01392-6.

IEA, 2023. World Energy Outlook 2023. International Energy Agency.

Islam, J., Vasant, P.M., Negash, B.M., et al., 2020. A holistic review on artificial intelligence techniques for well placement optimization problem. Adv. Eng. Software 141, 102767. https://doi.org/10.1016/j.advengsoft.2019.102767.

Kumar, A., 2021. Multi-stage, multi-swarm PSO for joint optimization of well placement and control. ArXiv Preprint ArXiv 2106, 01146. https://doi.org/10.48550/arXiv.2106.01146.

Levchenko, P., Iskakov, E., 2023. Simulation-based methodology for optimization of infill well locations in a mature oilfield. In: SPE Caspian Technical Conference and Exhibition 2023. https://doi.org/10.2118/217506-MS.

Li, X., Zhang, J., Chang, J., et al., 2023. Optimization of automatic well pattern deployment in high water-cut oilfield. J. Energy Resour. Technol. 145 (11), 112904. https://doi.org/10.1115/1.4062994.

Liu, Y., 2008. Performance analysis and optimal design for well patterns in anisotropic formations. Pet. Sci. 5 (3), 251–257. https://doi.org/10.1007/s12182-008-0040-2.

Mao, J., Ghahfarokhi, A.J., 2024. Intelligent multi-well placement optimization for CO_2-EOR and storage (CEORS) in complex reservoirs. ECMOR 2024, 1–19. https://doi.org/10.3997/2214-4609.202437060.

Nocedal, J., Wright, S.J., 2006. In: Numerical optimization, second ed. Springer, Berlin. https://doi.org/10.1007/978-0-387-40065-5.
Wang, H., Ciaurri, D.-E., Durlofsky, L.-J., et al., 2012. Optimal well placement under uncertainty using a retrospective optimization framework. SPE J. 17 (1), 112–121. https://doi.org/10.2118/141950-PA.
Wang, L., Yao, Y., Luo, X., et al., 2023. A critical review on intelligent optimization algorithms and surrogate models for conventional and unconventional reservoir production optimization. Fuel 350, 128826. https://doi.org/10.1016/j.fuel.2023.128826.
Wang, P., Wang, S., Zhang, H., et al., 2022. Research and practice on oil displacement law of different well pattern in offshore oilfield. Geosyst. Eng. 24 (5–6), 275–286. https://doi.org/10.1080/12269328.2022.2054865.
Wang, S., Chen, S., 2019. Integrated well placement and fracture design optimization for multi-well pad development in tight oil reservoirs. Comput. Geosci. 23 (1), 471–493. https://doi.org/10.1007/s10596-018-9795-8.
Wu, G., Ma, Y., Cao, Y., et al., 2025. Optimization of well patterns in offshore low-permeability thin interbedded reservoirs: A numerical simulation study in the Bozhong oilfield, China. Energies 18 (2), 285. https://doi.org/10.3390/en18020285.
Xu, J., Cheng, L., Ma, L., 2007. A predictive model of enhanced oil recovery by infill drilling and its application. Pet. Sci. 4 (3), 47–50. https://doi.org/10.1007/s12182-007-0008-7.
Xu, Z.X., Li, S.Y., Li, B.F., et al., 2020. A review of development methods and EOR technologies for carbonate reservoirs. Pet. Sci. 17 (4), 990–1013. https://doi.org/10.1007/s12182-020-00467-5.
Xue, L., Gu, S.H., Jiang, X.E., et al., 2021. Ensemble-based optimization of hydraulically fractured horizontal well placement in shale gas reservoir through Hough transform parameterization. Pet. Sci. 18 (3), 839–851. https://doi.org/10.1007/s12182-021-00560-3.
Zhan, J., Tian, Y., Fan, C., et al., 2022. Well pattern and well spacing optimization of large volume water injection in a low-permeability reservoir with pressure sensitivity. Geofluids 2022 (1), 8215223. https://doi.org/10.1155/2022/8215223.
Zhang, Q., Li, H., 2007. MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Comput. 11 (6), 712–731. https://doi.org/10.1109/TEVC.2007.892759.
Zhao, X.J., Wu, J.W., Zuo, S.L., et al., 2022. Effect and trend of well pattern infillings in Daqing Oilfield. Petrol. Geol. Recov. Eff. 29 (5), 141–146. https://doi.org/10.13673/j.cnki.cn37-1359/te.202108049.
Zhuang, X.Y., Wang, W.D., Su, Y.L., et al., 2025. Deep learning-assisted optimization for enhanced oil recovery and CO_2 sequestration considering gas channeling constraints. Pet. Sci. 22 (8), 3397–3417. https://doi.org/10.1016/j.petsci.2025.04.028.