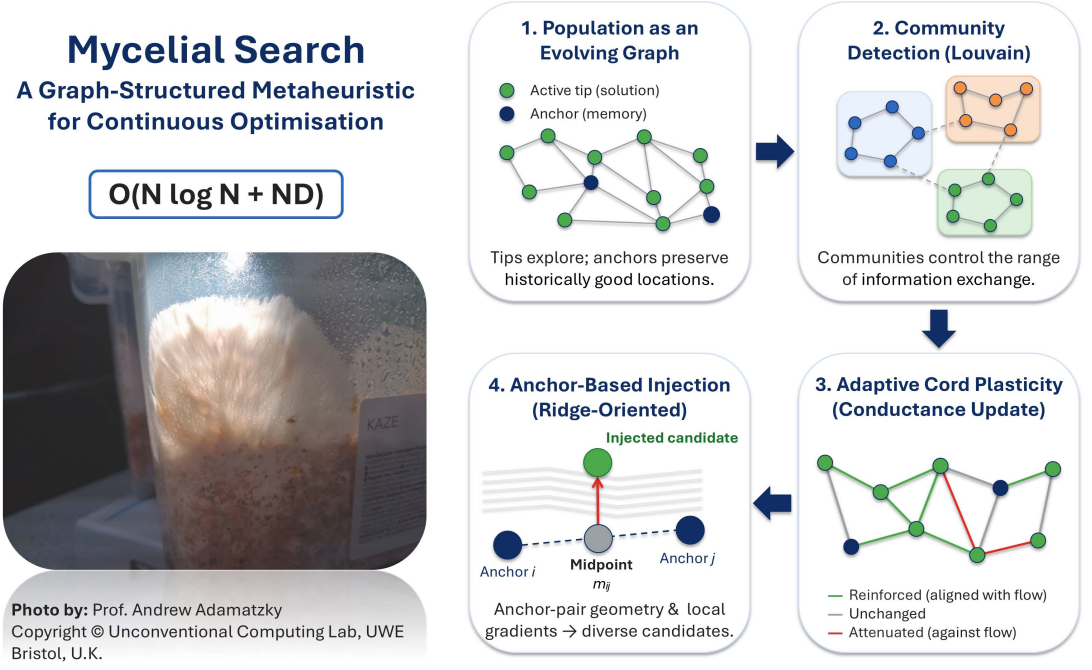
Graphical Abstract

Mycelial Search: A Graph-Structured Metaheuristic for Continuous Optimisation

Mohammad Mahdi Dehshibi



Highlights

Mycelial Search: A Graph-Structured Metaheuristic for Continuous Optimisation

Mohammad Mahdi Dehshibi

- Mycelial Search introduces an evolving graph structure for continuous optimisation.
- Community detection regulates information exchange across local search regions.
- Adaptive cord plasticity reinforces flow-aligned links and attenuates others.
- Per-iteration complexity scales as $O(N \log N + N D)$ under sparse graph connectivity.
- Myco achieves competitive results on CEC 2022 functions across two dimensions.

Mycelial Search: A Graph-Structured Metaheuristic for Continuous Optimisation

Mohammad Mahdi Dehshibi

Unconventional Computing Laboratory UWE Bristol UK

ARTICLE INFO

Keywords:
Continuous optimisation
Metaheuristic optimisation
Graph-structured search
Community detection
Adaptive conductance
Mycelia-inspired computing

ABSTRACT

Continuous optimisation methods need to balance sharing information and maintaining alternative search directions. In this paper, we introduce Mycelial Search (Myco), a graph-structured metaheuristic designed around active tips, community-weighted flow, adaptive cord plasticity, and anchor-based injection. Candidate solutions form an evolving spatial graph in which a Louvain partition distinguishes within-community from cross-community information exchange. Adaptive cord plasticity subsequently modifies active tip-to-tip edges according to their alignment with the local flow. An anchor-based injection mechanism supplements the graph-driven tip dynamics. We evaluated Myco on the CEC 2022 single-objective bound-constrained benchmark suite at dimensions $D = 10$ and $D = 20$, using 30 independent runs per algorithm-function pair. The comparison includes eleven established optimisers from several search families. Myco reaches competitive results on selected functions across both dimensions. The ablation analysis further shows that community structure regulates the range of graph-based information exchange, whereas cord plasticity controls the persistence of local directional influence. These findings indicate that graph-structured local interaction can support continuous optimisation, while its effectiveness depends on landscape structure and information transfer across local search regions.

1. Introduction

Continuous optimisation problems with interacting variables, multiple local optima, and regions with sharply different search behaviour appear in constrained engineering design and optimisation settings [10, 19, 24, 25, 30]. Population-based metaheuristics address this setting by maintaining a set of candidate solutions and using their relative quality to guide the search. Genetic algorithms use selection and variation, particle swarm optimisation uses learned exemplar positions, and differential evolution constructs trial solutions from population differences [13, 26, 31]. Later methods refined these principles through adaptive control parameters, external archives, and changing population sizes [5, 28, 33].

Many population methods still represent the relations among candidate solutions only weakly. Interaction is often mediated by a global best solution, sampled exemplars, or temporary population differences. These mechanisms can be effective, but they rarely retain a spatial interaction structure that changes with the search state. This matters when useful information is local. A candidate may benefit from nearby high-quality positions without requiring direct attraction to the same population-wide reference.

Graph-oriented optimisation offers a different perspective. Adaptive connectivity, flow, and edge reinforcement have been used to represent how information or resources move through a network [3, 11]. Community detection identifies groups that are more densely connected internally than externally [4]. These approaches commonly address problems in which a graph, route, or transport network is itself the object of optimisation. Candidate solutions in continuous search can be organised the same way as nodes in an evolving graph, where local connectivity determines which search information is exchanged and how strongly it contributes to subsequent motion.

Fungal mycelia offer a suitable biological motivation for this perspective. A mycelial network coordinates distributed exploration with transport across the colony. Established pathways can remain available while local connectivity adapts to usage patterns [1, 6, 7, 8, 9]. We use this principle without attempting to reproduce fungal physiology. We introduce Mycelial Search (Myco), a graph-structured metaheuristic for continuous optimisation. Myco represents candidate solutions as active tips and retains a bounded set of anchors at historically favourable

mohammad.dehshibi@uwe.ac.uk (M.M. Dehshibi)
ORCID(s):

positions within a spatial graph that is reconstructed at every iteration. Tips receive locally weighted flow information from neighbouring nodes. Anchors preserve high-quality locations that remain accessible to the evolving graph.

Myco combines two forms of graph organisation. A Louvain partition imposes a mesoscale organisation on the current graph and distinguishes within-community from cross-community interactions [4]. A conductance update then modifies the current tip-to-tip edges based on their directional agreement with the induced flow, reinforcing flow-aligned edges and attenuating unsupported ones. The community partition regulates the scope of information exchange across the graph. The conductance update retains short-term directional information within the local graph.

We evaluate Myco on the CEC 2022 [14] single-objective bound-constrained benchmark suite across two problem dimensions and compare it with established optimisation methods from several search families. We also examine the separate contributions of adaptive conductance and community partitioning. This analysis tests whether an evolving local interaction graph and adaptive edge strength provide useful search information in continuous optimisation.

The rest of this paper is organised as follows: Section 2 reviews population-based and graph-oriented optimisation. Section 3 presents Myco and its graph construction, community-weighted flow, adaptive conductance, and auxiliary search components. Section 4 describes the experimental protocol, comparative results, and ablation analyses. The final section concludes the paper and identifies limitations of the current design.

2. Related Work

Research on metaheuristic optimisation has produced a wide range of algorithms that may appear distinct at the metaphorical level, yet often share similar underlying computational structures [18, 23]. In many approaches, candidate solutions are updated using individual-level rules and exchange information through limited elite references or population-wide guidance, even as operator design and search configuration become more sophisticated [32]. Although such designs have led to many effective optimisers, they often provide only limited support for local cooperation, spatial grouping, and short-term structural memory. To address these limitations, research studies have increasingly explored methods that explicitly incorporate agent connectivity and population organisation into the search process, rather than relying solely on flat population-level updates [17].

2.1. Population-Based Search

Population-based optimisation is based on the idea that a set of concurrently updated candidate solutions can balance exploration and exploitation more effectively than a purely local, single-solution procedure. Genetic algorithm (GA) relies on population evolution through selection and variation [31]. Particle swarm optimisation (PSO) updates each particle by combining its own search history with information obtained from exemplar positions in the swarm [13]. Differential evolution (DE) generates trial solutions by taking vector differences among population members and has become a foundation of continuous global optimisation [26].

Subsequent work focused on increasing search quality by refining how individuals learn, mutate, or adapt their control parameters. CLPSO widened the learning source of each particle and improved diversity maintenance in multimodal search through comprehensive exemplar selection [16]. In the DE family, SAP-DE explored self-adaptive population sizing within differential evolution [29], JADE introduced adaptive differential evolution with an optional archive [33], SHADE formalised success-history-based parameter adaptation [27], L-SHADE added linear population size reduction [28], and jSO continued this progression toward high-performance real-parameter optimisation [5]. While these methods show that substantial gains can be obtained by improving parameter control, mutation design, and learning strategy within a shared population-update framework, their search logic remains centred on updating individuals within a flat population.

Bio-inspired optimisers reformulated the population search logic by emulating various natural processes. Artificial bee colony (ABC) modelled foraging behaviour through employed, onlooker, and scout bees [12]. The grey wolf optimiser (GWO) and the whale optimisation algorithm (WOA) describe the search process through leadership hierarchy and hunting behaviour [21, 22]. The slime mould algorithm (SMA) translates oscillatory adaptation in slime mould into a stochastic optimiser [15]. The moss growth optimisation (MGO) is a recent example that organises the search through wind-direction estimation, spore dispersal, dual propagation, and cryptobiosis [34]. Despite their differences in inspiration and update mechanisms, these algorithms still update a population whose relational structure is only weakly modelled. This observation is important for the present paper because it motivates the shift from a flat population of individually updated agents to graph-structured search, in which candidate solutions interact through an explicitly defined and evolving interaction graph.

2.2. Network- and Graph-Oriented Optimisation

A more structured alternative to flat population search appears in optimisation methods that operate through adaptive graphs. Instead of relying solely on isolated update rules, these approaches allow connectivity, flow balance, and path reinforcement to influence the evolution of the search over time [3]. Physarum-inspired optimisation is a representative case, since it converts adaptive transport behaviour into graph-based computational dynamics [11].

Such methods have mainly been developed for network-centred problems. The accelerated Physarum solver demonstrated that the number of inactive nodes can be reduced and computation can be terminated earlier without sacrificing effectiveness in network optimisation tasks [11]. Physarum-inspired routing methods have similarly used adaptive graph dynamics to reconstruct favourable transmission paths in changing communication environments [20].

While these studies show that evolving network structure can guide optimisation, they do not fully address the setting considered in this study. Graph-oriented methods were designed for problems where the solution is already a path, route, or transport network [3, 20]. Mycelial Search (Myco) is motivated by this gap. It adopts a graph-based perspective on adaptive connectivity and flow-based reinforcement to organise interactions among candidate solutions in continuous optimisation. Therefore, the evolving network becomes a search mechanism rather than only a problem-specific representation.

3. Proposed Method

This section presents the Mycelial Search (Myco) at three connected levels. We first state the biological motivation that guided the design. We then define the network construction, the community-driven flow mechanism, the cord plasticity rule, and the anchor and Ridge-Oriented Injection (ROI) updates. Finally, we summarise the full algorithm and its computational cost.

3.1. Biological Motivation

The proposed method is motivated by the way fungal mycelia coordinate exploration and transport through a distributed network [1]. Fungal mycelia solve two linked problems at once: expanding into new regions while maintaining transport across the already-formed network. This dual behaviour motivates our proposed Mycelial Search, which formulates search as a spatial network in which candidate solutions interact through local connections, directional flow, and adaptive edge strength.

Myco does not model chemotropic growth or substrate-level physiology. Instead, it borrows the idea of many local explorers moving concurrently through a shared networked search space. Tips represent active exploratory units. Anchors preserve historically strong locations and stabilise the evolving network. Together, these components reflect three recurring features of mycelial organisation: exploration remains distributed, promising locations remain accessible to the rest of the network, and information is exchanged through local neighbourhoods rather than a single global communication pattern.

The strongest biological link in the method is the conductance update on tip-to-tip connections. In fungal transport networks, frequently used pathways tend to persist, whereas weakly used pathways regress [2, 6, 8]. The proposed method implements the same directional idea through adaptive cord conductance. Edges aligned with flow are reinforced, while edges not supported by flow decay over time. This mechanism is central to the method, where the community structure enters at a different level. The Louvain partition [4] introduces an intermediate scale of organisation. It is biologically motivated only in the limited sense that real mycelial networks exhibit spatially organised domains. However, the partition itself is algorithmic rather than biological. ROI remains outside the main biological claim and is included only as an auxiliary search mechanism.

3.2. Network Construction

We represent the current search state as an undirected, weighted spatial graph $\mathcal{G}$ whose nodes consist of active tips and retained anchors. Let D denote the problem dimension, N the number of tips, and K the number of retained anchors, where $K \leq K_{\max}$. The position of tip i is denoted by $x_i \in \mathbb{R}^D$ for $i = 1, \dots, N$, and the position of anchor k is denoted by $a_k \in \mathbb{R}^D$ for $k = 1, \dots, K$. The corresponding node set is $\mathcal{V} = \{x_1, \dots, x_N, a_1, \dots, a_K\}$.

To keep the retained-anchor set compact, we cap the set at $K_{\max}$. This way, if more than $K_{\max}$ anchors are available, only the $K_{\max}$ anchors with the lowest objective values are retained.

Connectivity is induced locally through a fixed fusion radius. Let $\mathbf{l} \in \mathbb{R}^D$ and $\mathbf{u} \in \mathbb{R}^D$ denote the lower and upper bounds of the search space. We set the radius to $r_{\text{fuse}} = 0.1 \, \|\mathbf{u} - \mathbf{l}\|_2$. The radius criterion determines only whether an

edge is present. Since it does not specify the interaction strength between connected nodes, we quantify the effective edge weight using Eq. (1).

$$e_{pq} = \frac{g_{pq}}{\|z_p - z_q\|_2 + \varepsilon}, \quad \varepsilon = 10^{-6} \tag{1}$$

where g_{pq} denotes the stored conductance associated with the undirected pair (p, q), and z_p and z_q are the position of nodes p and q in $\mathcal{V}$

The conductance is shared symmetrically by (p, q) and (q, p). Previously unseen tip-to-tip pairs are initialised with $g_{pq} = 1$, whereas anchor-incident edges retain fixed conductance values and are not modified by plasticity. Pairs that fail the radius condition are simply absent from the graph at that iteration. In this construction, anchors serve as elite landmarks that shape the local neighbourhood structure and influence the subsequent flow field without themselves becoming adaptive cords. We restrict conductance adaptation to tip-to-tip edges so that anchors provide stable reference points while the search network remains free to reorganise around them.

3.3. Cord Plasticity

We use cord plasticity to provide a short-term memory mechanism on the weighted graph by reinforcing directions that are repeatedly supported by the induced flow. The rule applies only to present tip-to-tip edges in the current graph. Let $\mathcal{N}_i$ denote the set of all neighbours of tip i in the current graph, including anchors when present, and let $\mathcal{N}_i^{\text{tip}} \subseteq \mathcal{N}_i$ denote the subset containing only tip neighbours. For each undirected tip pair (i, j) with $j \in \mathcal{N}_i^{\text{tip}}$, the conductance is stored as a single shared scalar $g_{ij} = g_{ji}$. Edges incident to anchors remain part of the graph and contribute to the flow field, but they are excluded from conductance adaptation. Let $\phi_i \in \mathbb{R}^D$ be the flow vector associated with tip i, and τ_0 denote a numerical tolerance. For $\|\phi_i\|_2 \geq \tau_0$, we normalize the local flow according to Eq. (2).

$$\hat{\phi}_i = \frac{\phi_i}{\|\phi_i\|_2}. \tag{2}$$

To define the edge direction vector, we use $c_{ij} = x_j - x_i$ for each $j \in \mathcal{N}_i^{\text{tip}}$. If $\|c_{ij}\|_2 < \tau_0$, the edge is left unchanged during the update of tip i. Otherwise, directional agreement between the edge and the local flow is measured by a cosine alignment score defined in Eq. (3).

$$s_{ij} = \frac{c_{ij}^{\top} \hat{\phi}_i}{\|c_{ij}\|_2}. \tag{3}$$

Given the alignment score in Eq. (3), reinforcement is applied only when $s_{ij} > \tau_{\text{align}}$, where τ_{align} is the alignment threshold. In that case, the reinforced conductance is obtained from Eq. (4).

$$\tilde{g}_{ij} = \min\left(g_{\max},\, g_{ij}(1 + \eta s_{ij})\right), \tag{4}$$

where η is the reinforcement rate and $g_{\max}$ is the upper conductance bound. For $s_{ij} \leq \tau_{\text{align}}$, we set $\tilde{g}_{ij} = g_{ij}$. Decay with lower clipping is then applied through Eq. (5).

$$g_{ij}^{+} = \max\left(g_{\min},\, (1 - \delta)\tilde{g}_{ij}\right). \tag{5}$$

where δ is the decay rate and $g_{\min}$ is the lower conductance bound.

Because plasticity is evaluated once from the perspective of each tip, the same undirected conductance may be revised twice within a single iteration, once when processing endpoint i and once when processing endpoint j. Therefore, the final end-of-iteration value may include a second update from the opposite endpoint. This update structure preserves conductance symmetry because both endpoints modify the same shared scalar. As a result, edges repeatedly aligned with the induced flow are first reinforced using Eq. (4) and then decayed and clipped using Eq. (5), whereas in the low-flow regime only Eq. (5) is applied.

3.4. Community Structure and Mesoscale Flow

We use the Louvain method [4] to partition $\mathcal{G}$ into communities at each iteration. For all reported experiments, we fix the community-detection random seed to 42. The raw partition is refined by a two-criterion filter. A community is retained if it contains at least $n_{\min}$ nodes and its induced subgraph density satisfies $\rho \geq \rho_{\min}$, where $\rho = 2m/(n(n-1))$ for a subgraph with n nodes and m internal edges. Nodes belonging to rejected communities are each reassigned a unique singleton label. They remain in the graph with all edges preserved and continue to contribute to the flow computation on equal terms with all other nodes. After reassignment, interactions involving singleton-labelled nodes use λ_{inter} in Eq. (7).

We define a potential for every node $v \in \mathcal{V}$, over both tips and anchors. Let $f(v)$ denote the objective value associated with node v. With $f_{\min}$ and $f_{\max}$ computed over all nodes in $\mathcal{V}$ at the current iteration, the potential is defined in Eq. (6).

$$s(v) = \begin{cases} \frac{f_{\max} - f(v)}{f_{\max} - f_{\min} + \varepsilon}, & f_{\max} \neq f_{\min} \\ 1, & \text{otherwise} \end{cases}, \qquad \varepsilon = 10^{-6} \tag{6}$$

For tip i, the flow vector ϕ_i is defined as a weighted mean of spatial displacements toward all neighbours $j \in \mathcal{N}_i$ in the current graph, including anchors. We define the message weight from node j to tip i as in Eq. (7).

$$m_{ij} = \lambda(i,j) \cdot e_{ij} \cdot \sigma_{ij}, \tag{7}$$

where e_{ij} is the edge weight. The community factor $\lambda(i,j)$ takes the value λ_{intra} for node pairs in the same filtered community and λ_{inter} for node pairs in different filtered communities, where λ_{intra} and λ_{inter} are constant weights for within-community and cross-community interactions, respectively. The sigmoid gate is defined in Eq. (8).

$$\sigma_{ij} = \frac{1}{1 + \exp\big(-\kappa\,(s_j - s_i)\big)}, \tag{8}$$

where σ_{ij} modulates the contribution according to the potential difference between neighbouring node j and receiving tip i. We compute the flow vector ϕ_i as the weighted mean of spatial displacements toward the neighbours of tip i using Eq. (9).

$$\phi_i = \frac{\sum_{j \in \mathcal{N}_i} m_{ij}\,(z_j - z_i)}{\sum_{j \in \mathcal{N}_i} m_{ij}}, \tag{9}$$

where z_i and z_j are the position of node i and j in $\mathcal{V}$, respectively. When the total message weight is zero, $\phi_i = \mathbf{0}$. We anneal the sigmoid steepness κ linearly over the run. Let $\text{FE}(t)$ denote the cumulative number of function evaluations consumed up to the start of iteration t, and let $\text{FE}_{\max}$ be the total evaluation budget. Then Eq. (10) defines the linear annealing schedule for κ.

$$\kappa(t) = \kappa_{\max} - (\kappa_{\max} - \kappa_{\min})\,\frac{\text{FE}(t)}{\text{FE}_{\max}}. \tag{10}$$

A larger κ makes the sigmoid more selective with respect to potential differences, suppressing contributions from neighbours with lower potential. As κ decreases toward $\kappa_{\min}$, the gate becomes less selective and the flow aggregates more uniformly across the neighbourhood.

3.5. Tip Update and Anchor Management

We update the tip states and the retained-anchor set within the same iteration. We first evaluate the tips and update the incumbent best position $x^\star(t)$, which is the only position eligible for anchor insertion. We insert this position only when its Euclidean distance from every retained anchor exceeds $\varepsilon = 10^{-6}$. If the number of retained anchors exceeds

$K_{\max}$, we keep the anchors with the best stored objective values and reevaluate the retained anchors before computing the node-wise potentials. These updated anchor values are then used in the current potential field and the resulting flow computation. When ROI is active, we also use values to evaluate the relative fitness gap that triggers the mechanism.

Let $v_i^t \in \mathbb{R}^D$ denote the velocity of tip i at iteration t and $r_i^t \in [0,1]^D$ be a random vector sampled independently for tip i. We perturb the current velocity of tips with at most one graph neighbour before applying the main velocity update. This perturbation is defined in Eq. (11).

$$\tilde{v}_i^t = \begin{cases} v_i^t + \xi_i^t, & |\mathcal{N}_i| \le 1 \\ v_i^t, & |\mathcal{N}_i| > 1 \end{cases}, \quad \xi_i^t \sim \mathcal{U}([-\delta,\delta]^D), \quad \delta = 0.05\, r_{\text{fuse}}, \tag{11}$$

where $\mathcal{U}([-\delta,\delta]^D)$ denotes the uniform distribution over the hypercube $[-\delta,\delta]^D$. We then calculate the velocity of tip i at iteration $t+1$ using Eq. (12).

$$v_i^{t+1} = w\, \tilde{v}_i^t + c_{\text{exp}}\, r_i^t \odot (x^\star(t) - x_i^t) + c_{\text{flow}}\, \phi_i, \tag{12}$$

where w is the inertia coefficient, c_{exp} is the coefficient of the best-position term, c_{flow} is the coefficient of the flow term, and $\odot$ denotes element-wise multiplication. We update the tip position through Eq. (13).

$$x_i^{t+1} = x_i^t + v_i^{t+1}. \tag{13}$$

We clip each component of x_i^{t+1} to the corresponding interval defined by $\mathbf{l}$ and $\mathbf{u}$ after the position update.

3.6. Ridge-Oriented Injection Spawning

Ridge-Oriented Injection (ROI) is an auxiliary mechanism that injects a candidate position derived from the retained-anchor set. It is considered only when at least two anchors are present. We sample two distinct anchors a_{k_1} and a_{k_2} uniformly, without replacement, and evaluate their relative fitness gap using Eq. (14).

$$\gamma = \frac{|f(a_{k_1}) - f(a_{k_2})|}{\max_{1\le k\le K} f(a_k) - \min_{1\le k\le K} f(a_k) + \varepsilon}. \tag{14}$$

The ROI branch proceeds when $\gamma > \tau_{\text{roi}}$, where τ_{roi} is the relative-gap threshold. The midpoint $x_{\text{mid}} = (a_{k_1} + a_{k_2})/2$ serves as the base for the spawn construction. We estimate $\nabla f(x_{\text{mid}})$ by central finite differences, as specified in Eq. (15).

$$\big[\nabla f(x_{\text{mid}})\big]_d = \frac{f(x_{\text{mid}} + h\,\mathbf{e}_d) - f(x_{\text{mid}} - h\,\mathbf{e}_d)}{2h}, \quad d = 1,\dots,D, \tag{15}$$

where $\mathbf{e}_d$ denotes the d-th standard basis vector and $h = 10^{-5}\max\{|f(x_{\text{mid}})|, 1\}$ is the step size. In addition to the evaluation of $f(x_{\text{mid}})$ needed to set h, this sub-step requires $2D+1$ objective evaluations.

The geometric adjustment preserves only the gradient component orthogonal to the anchor axis $u = (a_{k_2} - a_{k_1}) / \|a_{k_2} - a_{k_1}\|_2$, and the transverse component is derived using Eq. (16).

$$\begin{aligned} p_\perp &= \hat{p} - (\hat{p}^\top u)\, u, \\ \hat{p} &= \frac{\nabla f(x_{\text{mid}})}{\|\nabla f(x_{\text{mid}})\|_2}, \end{aligned} \tag{16}$$

When either $\|a_{k_2} - a_{k_1}\|_2$ or $\|\nabla f(x_{\text{mid}})\|_2$ is below $\tau_{\text{deg}} = 10^{-9}$, the correction degenerates and x_{mid} is passed through unchanged. The spawn candidate is then calculated using Eq. (17).

$$x_{\text{roi}} = \Pi_{[\mathbf{l},\mathbf{u}]}\big(x_{\text{mid}} - \alpha_{\text{roi}}\, p_\perp\big), \tag{17}$$

where $\Pi_{[\mathbf{l},\mathbf{u}]}$ denotes component-wise clipping to the feasible domain and α_{roi} is the ROI correction coefficient. We evaluate x_{roi} and replace the current worst tip only when the spawned value is strictly lower, adding one objective evaluation to the cost of the triggered ROI branch.

3.7. Algorithm Summary and Complexity

Algorithm 1 summarises the complete procedure. We use it to consolidate the interaction between the retained-anchor mechanism, the community-weighted flow field, the conductance update, and the tip dynamics defined in Section 3.2 through Section 3.6. The resulting iteration uses the current tip and anchor evaluations to form the potential field, applies the filtered Louvain partition to the flow computation, updates tip-tip conductances based on the induced flow, and then advances the tips. When ROI is enabled, the same iteration may also inject one additional candidate derived from a selected anchor pair.

Algorithm 1. Mycelial Search

Require: Objective function f, dimension D, bounds $\mathbf{l}, \mathbf{u}$, number of tips N, anchor cap $K_{\max}$, evaluation budget $\mathrm{FE}_{\max}$, model hyperparameters
Ensure: Incumbent best position $x^\star(t)$ and, when needed, its objective value $f^\star(t)$

% Initialization %
$r_{\mathrm{fuse}} \leftarrow 0.1\|\mathbf{u} - \mathbf{l}\|_2$
Initialize $x_i \sim \mathcal{U}([\mathbf{l}, \mathbf{u}])$ and $v_i^0 \leftarrow \mathbf{0}$ for $i = 1, \ldots, N$
Initialize retained anchors $\mathcal{A} \leftarrow \emptyset$ and the conductance map for tip-tip pairs
Set $\mathrm{FE}(t) \leftarrow 0$, $f^\star(t) \leftarrow +\infty$, and $x^\star(t) \leftarrow$ null
while $\mathrm{FE}(t) < \mathrm{FE}_{\max}$ **do**

% Annealed Sigmoid Steepness %
    Update $\kappa(t)$ using Eq. (10) with the cumulative evaluation count at the start of the current iteration

% Tip Evaluation and Incumbent Update %
    Evaluate all current tips and increment $\mathrm{FE}(t)$ by N
    Update $f^\star(t)$ and $x^\star(t)$ if a better tip is found

% Anchor Management %
    Insert $x^\star(t)$ into $\mathcal{A}$ only if its Euclidean distance from every retained anchor exceeds 10^{-6}
    if $|\mathcal{A}| > K_{\max}$ **then**
        Retain the $K_{\max}$ anchors with the best stored objective values
    end if
    Reevaluate all retained anchors and increment $\mathrm{FE}(t)$ by $|\mathcal{A}|$

% Node-wise Potentials from Already Available Objective Values %
    Form the node set $\mathcal{V} = \{x_1, \ldots, x_N\} \cup \mathcal{A}$
    Collect the already evaluated objective values $\{f(v) : v \in \mathcal{V}\}$ from the current tips and retained anchors
    Compute $s(v)$ for all $v \in \mathcal{V}$ using Eq. (6)

% Graph Construction and Community Structure %
    Construct the current graph $\mathcal{G}$ using r_{fuse}
    Compute edge weights e_{pq} using Eq. (1)
    Compute the Louvain partition of $\mathcal{G}$ with seed 42
    Filter the raw communities using $n_{\min}$ and $\rho_{\min}$, then relabel rejected communities as singletons

% Mesoscale Flow %
    for $i \leftarrow 1$ to N **do**
        Compute ϕ_i from Eqs. (7)–(9)
    over all $j \in \mathcal{N}_i$, using the previously computed
    potentials and filtered community labels
    end for

% Cord Plasticity %
    for $i \leftarrow 1$ to N **do**

Update the conductances of present tip-tip edges (i, j) with $j \in \mathcal{N}_i^{\text{tip}}$ using the previously computed flow vector ϕ_i and Eqs. (2)–(5)
end for

% Tip Motion %
for $i \leftarrow 1$ to N **do**
if $|\mathcal{N}_i| \leq 1$ **then**
Sample $\xi_i^t \sim \mathcal{U}([-\delta, \delta]^D)$ with $\delta = 0.05\, r_{\text{fuse}}$
Set $\tilde{v}_i^t \leftarrow v_i^t + \xi_i^t$
else
Set $\tilde{v}_i^t \leftarrow v_i^t$
end if
Sample $r_i^t \sim \mathcal{U}([0, 1]^D)$
Update v_i^{t+1} using Eq. (12)
Update x_i^{t+1} using Eq. (13)
Clip x_i^{t+1} component-wise to $[\mathbf{l}, \mathbf{u}]$
end for

% Auxiliary ROI Branch %
if ROI is enabled and $|\mathcal{A}| \geq 2$ **then**
Sample two distinct anchors $a_{k_1}, a_{k_2} \in \mathcal{A}$ uniformly without replacement
Compute the relative gap γ using Eq. (14)
if $\gamma > \tau_{\text{roi}}$ **then**
Set $x_{\text{mid}} \leftarrow (a_{k_1} + a_{k_2})/2$
Estimate $\nabla f(x_{\text{mid}})$ using Eq. (15)
Update $\text{FE}(t) \leftarrow \text{FE}(t) + 2D + 1$
Compute $p_\perp$ using Eq. (16)
Construct x_{roi} using Eq. (17)
Evaluate x_{roi} and update $\text{FE}(t) \leftarrow \text{FE}(t) + 1$
Replace the current worst tip only if x_{roi} yields a strictly lower objective value
end if
end if
end while
return $x^\star(t)$ and $f^\star(t)$

Let $K \leq K_{\max}$ denote the number of retained anchors at the current iteration, let $M = N + K$ denote the total number of graph nodes, and let $E = |\mathcal{E}|$ denote the number of graph edges. Excluding the cost of objective-function evaluation, graph construction with spatial neighbourhood queries requires $O(M \log M + E)$ operations. The mesoscale flow computation and the cord-plasticity update each scale as $O(ED)$. The velocity and position updates scale as $O(ND)$. Therefore, the per-iteration arithmetic cost is given by Eq. (18).

$$O(M \log M + ED + ND). \tag{18}$$

We cap the anchor set by $K_{\max}$, and the fusion radius induces local connectivity. In the sparse regime considered here, E therefore grows approximately linearly with M. Under this condition, Eq. (18) reduces in practice to $O(N \log N + ND)$. Each iteration requires $N + K$ objective evaluations from tip evaluation and anchor reevaluation. When the ROI branch is triggered, it adds $2D + 2$ additional evaluations: $2D + 1$ for the midpoint-based numerical gradient and 1 for the spawned candidate. Since $K \leq K_{\max}$, the baseline evaluation cost per iteration remains linear in N, and the total number of evaluations is bounded by $\text{FE}_{\max}$.

Table 1
Experimental protocol for the CEC 2022 benchmark evaluation.

Protocol component	$D = 10$	$D = 20$
Test functions	F1–F11	F1–F11
Independent runs	30	30
Maximum function evaluations	200,000	1,000,000
Evaluation budget	$20{,}000D$	$50{,}000D$
Search bounds	$[-100, 100]$	$[-100, 100]$
Myco population	30 tips	30 tips
Comparator population	50 individuals	50 individuals

Table 2
External comparator set organised by algorithm family.

Algorithm family	Optimisers
Differential evolution	jSO [5], SAP-DE [29], L-SHADE [28], JADE [33]
Swarm intelligence	CLPSO [16], ABC [12]
Genetic	GA [31]
Mammal-inspired	GWO [22], WOA [21]
Decentralised growth	SMA [15], MGO [34]

This gives $O(\mathrm{FE}_{\max}/N)$ iterations and yields the total arithmetic cost $O\big(\mathrm{FE}_{\max}(\log N + D)\big)$, excluding the cost of the objective function itself. When D dominates $\log N$, the total arithmetic cost reduces to $O(\mathrm{FE}_{\max} D)$.

4. Experiments

We evaluate Myco on the CEC 2022 single-objective bound-constrained benchmark suite using functions F1 to F11 at dimensions $D = 10$ and $D = 20$. The experiments compare final-error performance, convergence behaviour, dimensional changes, and the contributions of cord plasticity and the community-detection backend. We first define the experimental protocol and comparator set. We then report the main comparative results and analyse the internal ablations.

4.1. Experimental Setup

Table 1 summarises the experimental protocol used for the CEC 2022 benchmark evaluation [14]. The experiments comprise 30 independent runs per algorithm and function. Myco achieved its reported results with 30 tips under an equal FE budget, while the comparator implementations used their configured population size of 50. We grouped the comparator methods by algorithmic family in Table 2. Final errors are reported by their mean and standard deviation. To facilitate reproducibility, the Python implementation of Myco is publicly available at `https://github.com/dehshibi/Mycelia-Search`.

Following the benchmark rule, random seeds were generated deterministically for the function f and run r by $I_{f,r} = fR + r - R$, followed by $\sigma_{f,r} = (I_{f,r} \bmod 1000) + 1$, where $R = 30$ is the number of independent runs. Convergence was recorded at 16 predefined checkpoints given by $\lfloor D^{k/5-3}\,\mathrm{FE}_{\max} \rfloor$ for $k = 0, \ldots, 15$, where $\mathrm{FE}_{\max}$ denotes the maximum number of function evaluations.

4.2. Results

Tables 3 and 4 report the final-error distributions at $D = 10$ and $D = 20$. The family ordering separates comparisons among methods that use different search principles. Boldface identifies the lowest mean final error for each function. Shading identifies the lowest mean within each non-differential-evolution family. The latter distinction is useful because a global row-wise winner does not, by itself, show whether the proposed network mechanism is competitive with search methods built on different information-sharing structures.

The benchmark classes impose distinct search demands. F1 is a shifted and fully rotated Zakharov function with a single basin. As a unimodal function, it tests directed progress in a rotated domain. F2 contains the curved and non-separable Rosenbrock valley. F3 to F5 present increasingly difficult multimodal conditions through expanded

Table 3: Final errors at $D = 10$ over 30 independent runs, reported as mean $\pm$ standard deviation. The reported error is the difference between the obtained objective value and the known optimum of the corresponding CEC 2022 function. Algorithms are grouped by search family. Boldface identifies the lowest mean final error for each function. Light shading identifies the lowest mean within the Growth Network, Genetic, Swarm Intelligence, Mammal-inspired, and Decentralised Growth families.

Function	Growth Network	Genetic/Evolutionary	Swarm Intelligence		Mammal-inspired		Decentralised Growth		Differential Evolution			
	Myco	GA	CLPSO	ABC	WOA	GWO	SMA	MGO	jSO	SAP-DE	L-SHADE	JADE
F1	**0 ± 0**	3494 ± 3337	1429 ± 486	4594 ± 1206	498 ± 1136	14.5 ± 19.7	$6.89 \times 10^{-4} \pm 3.45 \times 10^{-4}$	4900 ± 1797	**0 ± 0**	5635 ± 2189	**0 ± 0**	**0 ± 0**
F2	6.14 ± 2.43	7.34 ± 11.6	21 ± 13.8	8.74 ± 0.739	11.7 ± 9.38	7.37 ± 3.14	7.25 ± 1.87	60 ± 30.1	4.78 ± 3.53	**3.19 ± 2.98**	7.14 ± 2.6	6.45 ± 2.46
F3	3.61 ± 0.418	3.2 ± 0.374	3.12 ± 0.16	3.26 ± 0.148	3.54 ± 0.294	2.21 ± 0.493	2.43 ± 0.459	3.02 ± 0.289	2.71 ± 0.475	2.62 ± 0.315	2.34 ± 0.278	**1.8 ± 0.373**
F4	178 ± 737	70.4 ± 22.9	69.9 ± 9.05	40.1 ± 4.9	1565 ± 1073	26.8 ± 11.3	29.5 ± 8.8	903 ± 390	22.9 ± 7.76	22.6 ± 4.55	9.16 ± 4.55	**7.28 ± 3.38**
F5	524 ± 617	97.6 ± 120	26 ± 11.3	8.93 ± 3.66	566 ± 343	1.54 ± 4.44	0.0727 ± 0.154	158 ± 63.6	0.0966 ± 0.172	7.83 ± 6.29	**0 ± 0**	**0 ± 0**
F6	92 ± 143	7034 ± 5061	$3.08 \times 10^4 \pm 8980$	428 ± 268	4066 ± 4026	3295 ± 2482	3184 ± 2124	$1.51 \times 10^4 \pm 4851$	3.55 ± 1.7	205 ± 380	**0.309 ± 0.218**	0.369 ± 0.111
F7	1691 ± 4411	2837 ± 3084	$4.00 \times 10^4 \pm 2.55 \times 10^4$	2472 ± 605	$3.50 \times 10^4 \pm 1.86 \times 10^5$	3431 ± 2826	1614 ± 1054	$9.39 \times 10^4 \pm 8.30 \times 10^4$	20.2 ± 84.6	240 ± 313	**2.57 ± 13**	41.9 ± 125
F8	20.1 ± 0.233	6.34 ± 8.94	20.4 ± 0.542	20.2 ± 0.525	20 ± 0.156	18.3 ± 6.1	19.3 ± 3.59	19.8 ± 0.866	12.5 ± 9.4	0.54 ± 1.87	**0.057 ± 0.0661**	11.9 ± 7.03
F9	3.97 ± 2.9	346 ± 639	$8.46 \times 10^4 \pm 2.77 \times 10^4$	**0.9 ± 0**	4.86 ± 2.84	$2.68 \times 10^4 \pm 2.80 \times 10^4$	0.907 ± 0.012	$1.62 \times 10^7 \pm 1.26 \times 10^7$	1.7 ± 2.06	6.72 ± 16.1	**0.9 ± 0**	**0.9 ± 0**
F10	19.8 ± 19.3	298 ± 340	1466 ± 766	25.6 ± 11.3	54.7 ± 55	670 ± 2306	26.7 ± 19.6	$3.23 \times 10^7 \pm 2.73 \times 10^7$	7.64 ± 11.7	22.9 ± 48.9	**1.8 ± 0**	**1.8 ± 0**
F11	4.69 ± 1.78	1.93 ± 0.641	5805 ± 9212	26.8 ± 7.06	48.8 ± 27.8	10.7 ± 4.94	0.804 ± 0.294	$5.63 \times 10^{11} \pm 5.11 \times 10^{11}$	2.03 ± 0.845	3.3 ± 3.9	0.779 ± 0.165	**0.775 ± 0.147**

Table 4: Final errors at $D = 20$ over 30 independent runs, reported as mean $\pm$ standard deviation. The reported error is the difference between the obtained objective value and the known optimum of the corresponding CEC 2022 function. Algorithms are grouped by search family. Boldface identifies the lowest mean final error for each function. Light shading identifies the lowest mean within the Growth Network, Genetic, Swarm Intelligence, Mammal-inspired, and Decentralised Growth families.

Function	Growth Network	Genetic/Evolutionary	Swarm Intelligence		Mammal-inspired		Decentralised Growth		Differential Evolution			
	Myco	GA	CLPSO	ABC	WOA	GWO	SMA	MGO	jSO	SAP-DE	L-SHADE	JADE
F1	**0 ± 0**	2147 ± 972	3273 ± 895	$2.68 \times 10^{4} \pm 3882$	7.89 ± 42.5	167 ± 230	$1.66 \times 10^{-3} \pm 8.47 \times 10^{-4}$	$2.06 \times 10^{4} \pm 2811$	**0 ± 0**	$2.31 \times 10^{4} \pm 7685$	**0 ± 0**	**0 ± 0**
F2	**29.9 ± 22.7**	53.6 ± 17.4	74.5 ± 9.44	79.3 ± 4.42	48.8 ± 21.6	56.7 ± 20.9	48.4 ± 4.6	227 ± 32.1	31.1 ± 22.2	32.5 ± 13.4	42.4 ± 16.6	48.8 ± 1.04
F3	8.11 ± 0.421	7.57 ± 0.477	7.5 ± 0.206	7.87 ± 0.133	8.07 ± 0.369	6.02 ± 0.571	6.47 ± 0.538	7.74 ± 0.29	7.26 ± 0.57	6.94 ± 0.36	6.75 ± 0.308	**5.81 ± 0.525**
F4	3775 ± 3935	225 ± 60.8	145 ± 14.3	233 ± 18.6	$1.07 \times 10^{4} \pm 3544$	128 ± 54.2	87.2 ± 17.6	6356 ± 1595	91.7 ± 18	101 ± 15.4	35.3 ± 5.48	**29.9 ± 9.93**
F5	2654 ± 1429	477 ± 239	138 ± 54.1	1005 ± 150	2392 ± 1107	27.4 ± 57.6	20.5 ± 34.8	838 ± 225	61.1 ± 70.7	467 ± 140	**0 ± 0**	$2.98 \times 10^{-3} \pm 0.0161$
F6	$9.53 \times 10^{4} \pm 4.66 \times 10^{4}$	$4.94 \times 10^{4} \pm 2.25 \times 10^{4}$	$3.90 \times 10^{5} \pm 1.31 \times 10^{5}$	$1.01 \times 10^{7} \pm 4.19 \times 10^{6}$	$1.84 \times 10^{4} \pm 1.16 \times 10^{4}$	$7.20 \times 10^{4} \pm 2.31 \times 10^{4}$	$1.47 \times 10^{5} \pm 2.71 \times 10^{4}$	$3.35 \times 10^{4} \pm 6176$	1962 ± 1062	$4.04 \times 10^{4} \pm 8783$	**1118 ± 534**	$1.23 \times 10^{4} \pm 1.21 \times 10^{4}$
F7	2144 ± 768	$5.66 \times 10^{4} \pm 6.28 \times 10^{4}$	$1.59 \times 10^{5} \pm 6.25 \times 10^{4}$	$1.52 \times 10^{6} \pm 4.50 \times 10^{5}$	$1.91 \times 10^{5} \pm 8.20 \times 10^{5}$	$1.79 \times 10^{5} \pm 1.91 \times 10^{5}$	9700 ± 4999	$1.68 \times 10^{6} \pm 3.18 \times 10^{6}$	557 ± 393	$8.12 \times 10^{4} \pm 1.21 \times 10^{5}$	**225 ± 316**	375 ± 428
F8	20.2 ± 0.412	**19.4 ± 3.54**	21.3 ± 0.14	21.5 ± 0.217	20.3 ± 0.397	21 ± 0.572	20.1 ± 0.0632	20.1 ± 0.0258	20 ± 0.0263	$20 \pm 9.32 \times 10^{-3}$	20 ± 0.0156	20.6 ± 0.161
F9	6.98 ± 3.85	37.2 ± 42.3	$6.93 \times 10^{4} \pm 2.17 \times 10^{4}$	$3.28 \times 10^{6} \pm 9.40 \times 10^{5}$	10.4 ± 5.36	$3.17 \times 10^{5} \pm 8.96 \times 10^{5}$	4.23 ± 3.32	$3.90 \times 10^{7} \pm 2.54 \times 10^{7}$	6.02 ± 4.05	9.76 ± 11.4	2.77 ± 2.23	**2.11 ± 1.16**
F10	57.6 ± 41.9	356 ± 398	3253 ± 1363	$1.33 \times 10^{6} \pm 4.37 \times 10^{5}$	122 ± 69.9	$2.08 \times 10^{4} \pm 5.19 \times 10^{4}$	34.4 ± 24.3	$5.33 \times 10^{7} \pm 5.12 \times 10^{7}$	75.5 ± 46.8	152 ± 337	10.1 ± 10.3	**7.44 ± 7.3**
F11	17.8 ± 9.52	**1.54 ± 0.0587**	$6080 \pm 1.03 \times 10^{4}$	$6.22 \times 10^{10} \pm 3.39 \times 10^{10}$	132 ± 48.8	$5.02 \times 10^{6} \pm 2.47 \times 10^{7}$	1.62 ± 0.533	$7.09 \times 10^{12} \pm 7.76 \times 10^{12}$	16.5 ± 9.56	10.8 ± 9.46	1.92 ± 0.722	1.61 ± 0.565

Schaffer, non-continuous Rastrigin, and Levy structures. F6 to F8 are hybrid functions composed of transformed subcomponents, while F9-F11 are composition functions with heterogeneous local regions [14]. Thus, the transition from F1 and F2 to the remaining classes shifts the principal challenge from locating a single favourable direction to maintaining search diversity while discriminating among multiple basins.

The exact convergence of Myco on F1 at both dimensions suggests that its spatial graph construction, anchor retention, and flow-driven tip motion are sufficient to reach the single-basin optimum within the present evaluation budgets. Its performance on F2 at $D = 20$ further indicates that the interaction network provides each tip with a locally weighted displacement field, while anchors preserve previously favourable locations. Together, these mechanisms form a local structural memory that can maintain multiple directional signals rather than forcing the population to collapse prematurely toward a single global reference.

The basic multimodal functions place greater pressure on cross-basin exploration and late-stage coordinate refinement. Their local minima, discontinuities, and mixed components require sustained movement among competing basins. The DE variants often perform well on these functions because their mutation operators use pairwise population differences and adapt control parameters based on successful runs. This mechanism can generate broad exploratory displacements before concentrating the search around promising regions. Among these variants, L-SHADE further reduces the population size, which can shift the search toward exploitation as the run progresses [28]. In contrast, Myco retains a local interaction graph. When this graph separates into spatially distinct groups, lower inter-community interaction weights reduce the influence of distant regions. This can preserve local organisation, but it may also delay the transfer of information needed to escape deceptive basins or to cross discontinuous boundaries.

The hybrid and composition functions test whether search information remains useful when several component landscapes compete across the domain. Myco can maintain competitive errors as long as the local flow field continues to identify a productive search direction. However, its performance weakens when relevant information is distributed across disconnected or competing regions of the landscape. The results suggest a trade-off in the current implementation: community-weighted local flow supports organised exploration, but it may also restrict rapid redistribution of search effort when the landscape begins to favour a distant component.

Figure 1 illustrates the temporal pattern of this distinction. While Figure 2 provides a median-based summary that is less sensitive to extreme runs, Figure 3 isolates the change in mean final error, showing that increasing the dimension alters the final-error profile of Myco in a consistent direction.

4.3. Ablation Study

We examine the respective contributions of adaptive cord conductance and the community-detection backend (i.e., Louvain and Greedy Modularity) within two ablation variants. Myco (Louvain) preserves the Louvain partition and all remaining components of the Plasticity, but disables the conductance update on tip-to-tip edges. Myco (Greedy) instead replaces the Louvain backend with Greedy Modularity optimisation. These comparisons isolate distinct aspects of the search graph: the former evaluates whether edge strengths should adapt to the induced flow, whereas the latter examines whether the resulting flow field benefits from a non-trivial mesoscale partition.

4.3.1. Cord Plasticity

Table 5 compares the Louvain and Plasticity variants using paired mean and standard-deviation values. The logarithmic ratio quantifies both the direction and magnitude of each paired difference, with negative values indicating improved performance under Plasticity.

At $D = 10$, Plasticity achieves its largest reductions on F4 and F6, with corresponding decreases in standard deviation. In the Louvain backend, existing edges are weighted according to distance, conductance, potential difference, and community membership, whereas tip-to-tip conductances remain fixed throughout the search. By contrast, Plasticity changes this local graph after the flow field has been computed. Such changes reinforce tip-to-tip edges that are aligned with the current flow while attenuating unsupported edges. The observed reduction in both location and dispersion suggests that adaptive conductance suppresses transient local directions that would otherwise retain excessive influence.

The effect becomes pronounced at $D = 20$. Plasticity reduces the mean error on F1, F3, F4, F5, F7, F8, F9, and F11. The paired reductions on F4 and F7 are associated with substantially lower standard deviations. This pattern suggests that the conductance update helps preserve locally persistent search directions when distance-based neighbourhoods become less informative in higher-dimensional landscapes. However, the benefit is not universal.

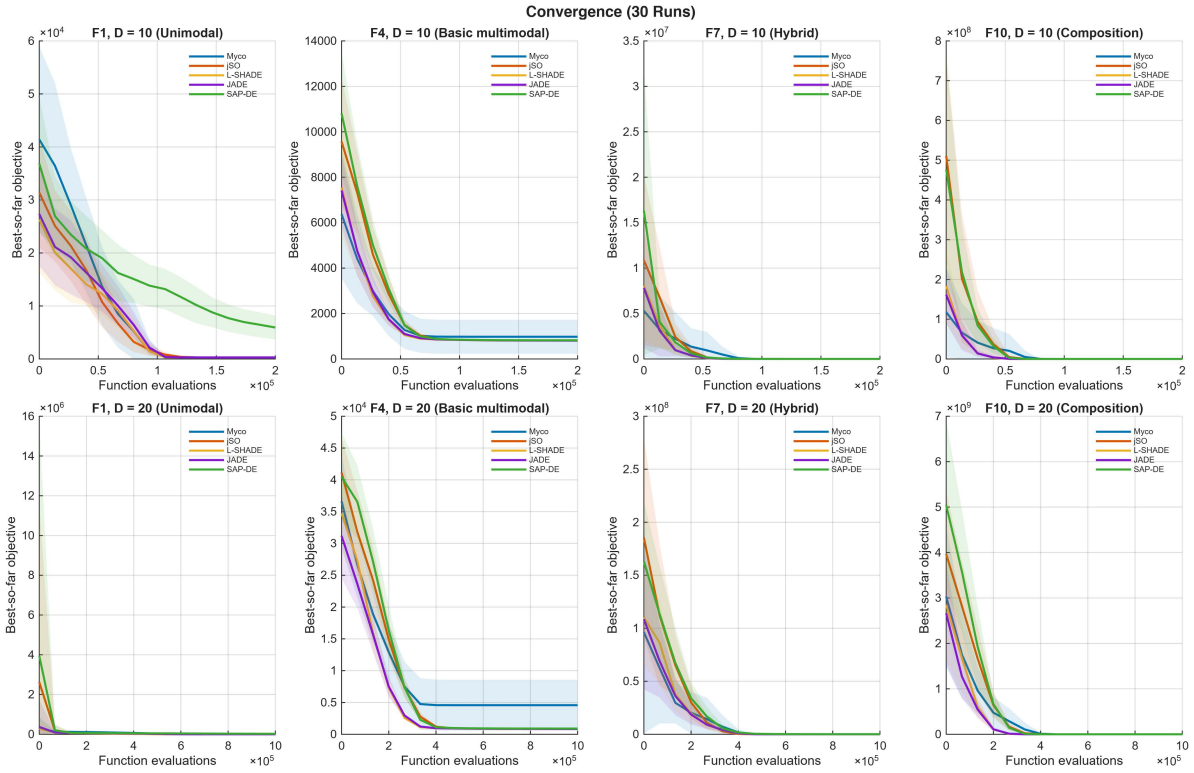


Figure 1: Mean best-so-far error trajectories over 30 runs for representative functions from the unimodal, basic multimodal, hybrid, and composition classes at $D = 10$ and $D = 20$. Shaded regions denote one standard deviation.

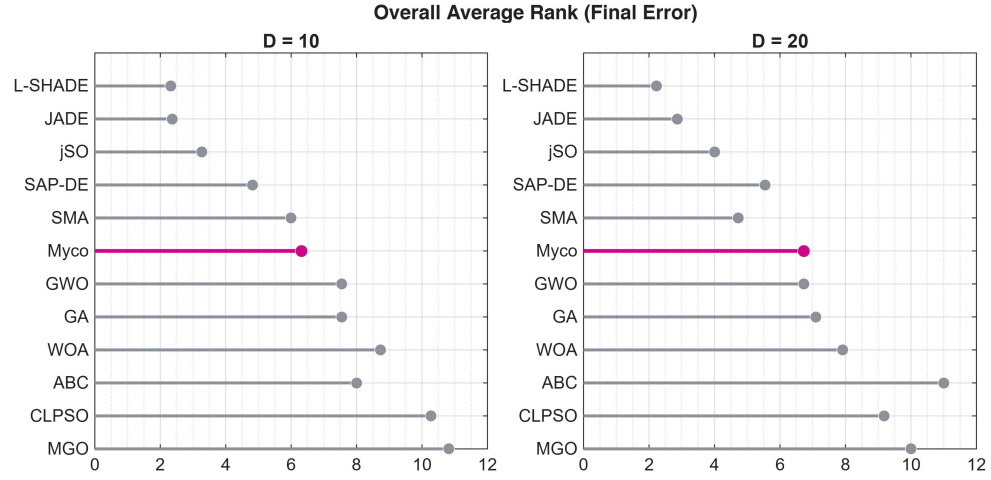


Figure 2: Average rank of each algorithm over F1 to F11 at $D = 10$ and $D = 20$, computed from median final error. Lower ranks indicate lower median error.

Louvain remains preferable on F2, F6, and F10, indicating that conductance decay can, in some landscapes, weaken alternative local directions before their search value becomes evident.

Figure 4 shows the paired final-error outcomes, whereas Figure 5 demonstrates the corresponding run-to-run variability. Table 5 complements these visual summaries by providing the exact relative changes that distinguish modest paired differences from order-of-magnitude reductions.

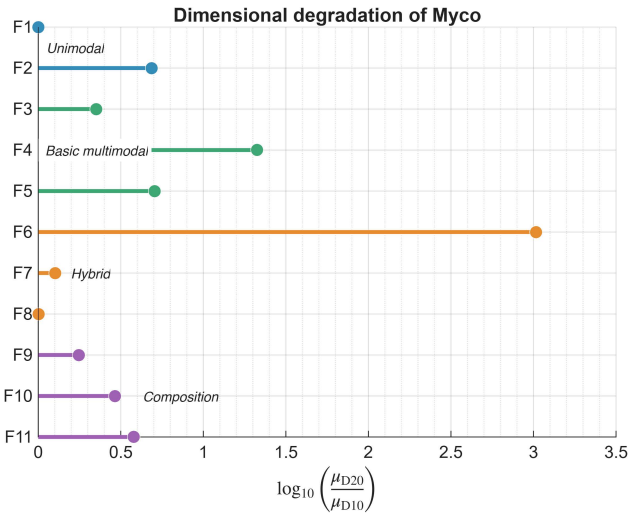


Figure 3: Change in the mean final error of Myco from $D = 10$ to $D = 20$ over F1 to F11, calculated as $\log_{10}(\mu_{D20}/\mu_{D10})$. Positive values indicate an increase in mean error at $D = 20$. Functions with zero mean error in both dimensions are assigned a value of zero.

Table 5
Effect of cord plasticity relative to the Louvain baseline across 30 independent runs. Values are reported as mean ± standard deviation. The final columns provide the relative effect size, computed as $\log_{10}(\mu_{\text{Plasticity}}/\mu_{\text{Louvain}})$, where negative values indicate lower mean error under Plasticity and positive values indicate lower mean error under Louvain. Empty entries report cases where one or both means are zero.

Function	$D = 10$			$D = 20$		
	Louvain	Plasticity	$\log_{10}\left(\frac{\mu_{\text{Plasticity}}}{\mu_{\text{Louvain}}}\right)$	Louvain	Plasticity	$\log_{10}\left(\frac{\mu_{\text{Plasticity}}}{\mu_{\text{Louvain}}}\right)$
F1	0 ± 0	0 ± 0	–	$4.95 \times 10^{-6} \pm 1.47 \times 10^{-5}$	0 ± 0	–
F2	5.67 ± 2.57	6.14 ± 2.43	0.034	25.6 ± 23.7	29.9 ± 22.7	0.066
F3	3.52 ± 0.382	3.61 ± 0.418	0.011	8.23 ± 0.500	8.11 ± 0.421	−0.007
F4	727 ± 1341	178 ± 737	−0.610	6848 ± 5127	3775 ± 3935	−0.259
F5	551 ± 709	524 ± 617	−0.021	3291 ± 1698	2654 ± 1429	−0.093
F6	865 ± 2305	92.0 ± 143	−0.973	$9.12 \times 10^4 \pm 3.94 \times 10^4$	$9.53 \times 10^4 \pm 4.66 \times 10^4$	0.019
F7	3110 ± 5507	1691 ± 4411	−0.265	3663 ± 4335	2144 ± 768	−0.233
F8	20.10 ± 0.166	20.09 ± 0.233	−0.000	20.21 ± 0.398	20.18 ± 0.412	−0.001
F9	3.17 ± 2.99	3.97 ± 2.90	0.097	7.79 ± 4.41	6.98 ± 3.85	−0.048
F10	16.3 ± 21.7	19.8 ± 19.3	0.083	45.4 ± 23.7	57.6 ± 41.9	0.103
F11	4.68 ± 3.37	4.69 ± 1.78	0.001	30.4 ± 14.5	17.8 ± 9.52	−0.233

4.3.2. Community Backend Sensitivity

Table 6 and Figure 6 report the sensitivity of the flow field to the choice of community detection backend. Across all reported functions and dimensions, the Greedy Modularity identifies a single raw community and a single retained community. Consequently, the distinction between intra-community and inter-community interactions is lost, and all present edges receive the same community factor.

The Greedy results contain extreme mean-to-median divergence for selected functions. At $D = 10$, this behaviour is most evident on F1, F9, F10, and F11, whereas at $D = 20$ it is most pronounced on F6, F7, F10, and F11. The low medians observed in several cases indicate that the one-community configuration can occasionally reach favourable regions of the search space. However, the substantially larger means suggest that a subset of runs enters poor regions from which recovery is not achieved within the available evaluation budget.

Louvain restores a partitioned flow field in which cross-community messages are weighted less strongly than within-community messages. Plasticity subsequently adjusts the relative strength of the remaining tip-to-tip interactions. These mechanisms therefore operate at distinct graph scales: Louvain regulates communication among local groups, whereas Plasticity controls the persistence of search directions within the current local graph.

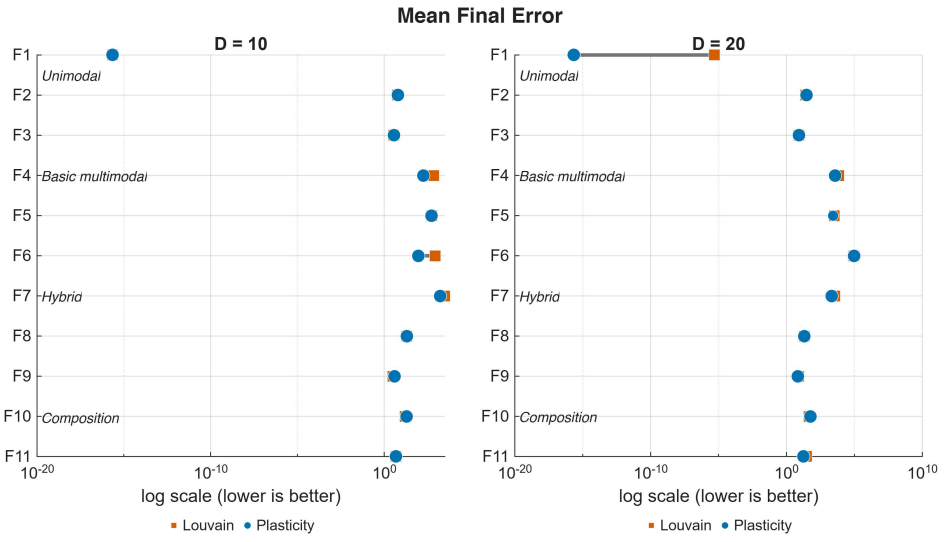


Figure 4: Mean final errors of Myco (Louvain) and Myco (Plasticity) over F1 to F11 at $D = 10$ and $D = 20$. Lower values are better. The horizontal axis is logarithmic.

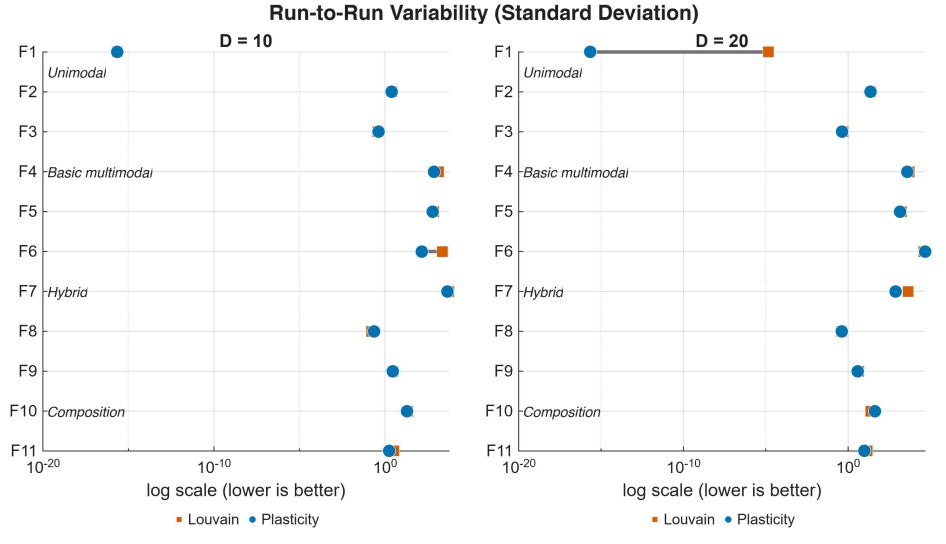


Figure 5: Standard deviations of final error for Myco (Louvain) and Myco (Plasticity) over 30 independent runs. Lower values indicate lower run-to-run variability. The horizontal axis is logarithmic.

5. Conclusion

In this study, we introduced Mycelial Search (Myco) to frame continuous optimisation as a search over an evolving local interaction graph. Candidate solutions are represented as active tips, while a bounded set of anchors preserves historically favourable positions. The graph itself is reconstructed at each iteration. We used the Louvain community detection algorithm to control the strength of information exchange within and across communities. We introduced cord plasticity, a rule that reinforces tip-to-tip connections aligned with the local flow and attenuates unsupported connections.

We evaluated Myco on the CEC 2022 single-objective bound-constrained benchmark suite, compared it with the established metaheuristic methods, and examined the roles of cord plasticity and the community detection backend in an ablation study. The ablation study results demonstrated that these mechanisms control different aspects of the search. Cord plasticity operates at the edge level, reinforcing or decaying individual tip-to-tip connections based on

Table 6
Greedy Modularity diagnostic for functions with large divergence between the mean and median final errors. Each entry reports 30 independent runs. The ratio $\mu/\tilde{x}$ indicates the extent to which rare high-error runs influence the mean.

Dimension	Function	Mean (μ)	Median ($\tilde{x}$)	$\mu/\tilde{x}$
$D = 10$	F1	2217	3.15×10^{-2}	7.05×10^{4}
$D = 10$	F7	2.99×10^{5}	6239.5	48.0
$D = 10$	F9	5.31×10^{6}	6.07	8.76×10^{5}
$D = 10$	F10	6.44×10^{6}	1146.8	5.62×10^{3}
$D = 10$	F11	2.09×10^{10}	4.16	5.01×10^{9}
$D = 20$	F6	7.81×10^{6}	1.03×10^{5}	75.9
$D = 20$	F7	6.83×10^{6}	1.83×10^{5}	37.4
$D = 20$	F10	3.65×10^{7}	3.70×10^{6}	9.86
$D = 20$	F11	7.72×10^{13}	48.5	1.59×10^{12}

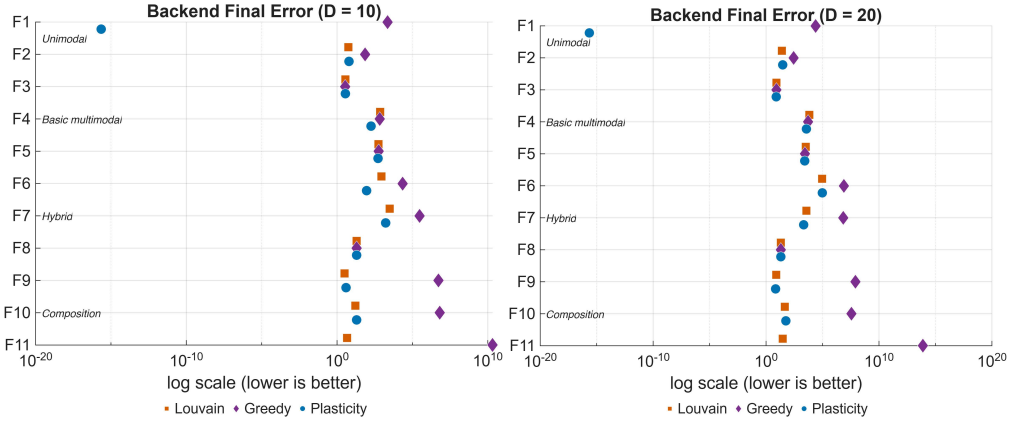


Figure 6: Mean final errors of the Louvain, Greedy Modularity, and Plasticity implementations over F1 to F11 at $D = 10$ and $D = 20$. Lower values are better. The horizontal axis is logarithmic.

their alignment with the local flow. The community detection backend operates at the partition level, determining the range over which information is exchanged.

Myco makes the interaction structure among candidate solutions explicit and subject to analysis during the search. The local graph is not only a mechanism for selecting neighbours. Its community partition and tip-to-tip edge conductances form part of the search state and affect how information is passed between candidate solutions. This formulation makes local organisation a component of continuous optimisation that can be defined, examined, and modified.

Acknowledgments

The author declares no competing interests.

References

[1] Adamatzky, A., Ayres, P., Beasley, A.E., Chiolerio, A., Dehshibi, M.M., Gandia, A., Albergati, E., Mayne, R., Nikolaidou, A., Roberts, N., Tegelaar, M., Tsompanas, M.A., Phillips, N., Wösten, H.A., 2022. Fungal electronics. Biosystems 212, 104588. URL: https://doi.org/10.1016/j.biosystems.2021.104588, doi:10.1016/j.biosystems.2021.104588.

[2] Adamatzky, A., Nikolaidou, A., Gandia, A., Chiolerio, A., Dehshibi, M.M., 2023. Reactive Fungal Wearable. Springer Nature Switzerland. chapter 8. pp. 93–104. URL: https://doi.org/10.1007/978-3-031-38336-6_8, doi:10.1007/978-3-031-38336-6_8.

[3] Awad, A., Coghill, G.M., Pang, W., 2023. A novel physarum-inspired competition algorithm for discrete multi-objective optimisation problems. Soft Computing 27, 14699–14719. URL: https://doi.org/10.1007/s00500-023-08505-1, doi:10.1007/s00500-023-08505-1.

[4] Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E., 2008. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment 2008, P10008. URL: https://doi.org/10.1088/1742-5468/2008/10/P10008, doi:10.1088/1742-5468/2008/10/P10008.

[5] Brest, J., Maučec, M.S., Bošković, B., 2017. Single objective real-parameter optimization: Algorithm jso, in: 2017 IEEE Congress on Evolutionary Computation (CEC), IEEE. pp. 1311–1318. URL: https://doi.org/10.1109/CEC.2017.7969456, doi:10.1109/CEC.2017.7969456.

[6] Dehshibi, M.M., Adamatzky, A., 2021. Electrical activity of fungi: Spikes detection and complexity analysis. Biosystems 203, 104373. URL: https://doi.org/10.1016/j.biosystems.2021.104373, doi:10.1016/j.biosystems.2021.104373.

[7] Dehshibi, M.M., Adamatzky, A., 2023. Complexity of Electrical Spiking of Fungi. Springer Nature Switzerland. chapter 4. pp. 33–60. URL: https://doi.org/10.1007/978-3-031-38336-6_4, doi:10.1007/978-3-031-38336-6_4.

[8] Dehshibi, M.M., Chiolerio, A., Nikolaidou, A., Mayne, R., Gandia, A., Ashtari-Majlan, M., Adamatzky, A., 2021. Stimulating Fungi Pleurotus ostreatus with Hydrocortisone. ACS Biomaterials Science & Engineering 7, 3718–3726. URL: https://doi.org/10.1021/acsbiomaterials.1c00752, doi:10.1021/acsbiomaterials.1c00752.

[9] Dehshibi, M.M., Chiolerio, A., Nikolaidou, A., Mayne, R., Gandia, A., Ashtari-Majlan, M., Adamatzky, A., 2023. On Stimulating Fungi Pleurotus Ostreatus with Hydrocortisone. Springer Nature Switzerland. chapter 9. pp. 105–121. URL: https://doi.org/10.1007/978-3-031-38336-6_9, doi:10.1007/978-3-031-38336-6_9.

[10] Dehshibi, M.M., Sourizaei, M., Fazlali, M., Talaee, O., Samadyar, H., Shanbehzadeh, J., 2017. A hybrid bio-inspired learning algorithm for image segmentation using multilevel thresholding. Multimedia Tools and Applications 76, 15951–15986. URL: https://doi.org/10.1007/s11042-016-3891-3, doi:10.1007/s11042-016-3891-3.

[11] Gao, C., Zhang, X., Yue, Z., Wei, D., 2020. An Accelerated Physarum Solver for Network Optimization. IEEE Transactions on Cybernetics 50, 765–776. URL: https://doi.org/10.1109/TCYB.2018.2872808, doi:10.1109/TCYB.2018.2872808.

[12] Karaboga, D., Basturk, B., 2007. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of Global Optimization 39, 459–471. URL: https://doi.org/10.1007/s10898-007-9149-x, doi:10.1007/s10898-007-9149-x.

[13] Kennedy, J., Eberhart, R., 1995. Particle swarm optimization, in: Proceedings of ICNN'95 - International Conference on Neural Networks, IEEE. pp. 1942–1948. URL: https://doi.org/10.1109/ICNN.1995.488968, doi:10.1109/ICNN.1995.488968.

[14] Kumar, A., Price, K.V., Mohamed, A.W., Hadi, A.A., Suganthan, P.N., 2022. Problem Definitions and Evaluation Criteria for the CEC 2022 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization. Technical Report. Nanyang Technological University. URL: https://github.com/P-N-Suganthan/2022-SO-BO/blob/main/CEC2022%20TR.pdf.

[15] Li, S., Chen, H., Wang, M., Heidari, A.A., Mirjalili, S., 2020. Slime mould algorithm: A new method for stochastic optimization. Future Generation Computer Systems 111, 300–323. URL: https://doi.org/10.1016/j.future.2020.03.055, doi:10.1016/j.future.2020.03.055.

[16] Liang, J., Qin, A., Suganthan, P., Baskar, S., 2006. Comprehensive learning particle swarm optimizer for solving multiobjective optimization problems. International Journal of Intelligent Systems 21, 209–226. URL: https://doi.org/10.1002/int.20128, doi:10.1002/int.20128.

[17] Ma, H., Shen, S., Yu, M., Yang, Z., Fei, M., Zhou, H., 2019. Multi-population techniques in nature inspired optimization algorithms: A comprehensive survey. Swarm and Evolutionary Computation 44, 365–387. URL: https://doi.org/10.1016/j.swevo.2018.04.011, doi:10.1016/j.swevo.2018.04.011.

[18] Ma, Z., Wu, G., Suganthan, P.N., Song, A., Luo, Q., 2023. Performance assessment and exhaustive listing of 500+ nature-inspired metaheuristic algorithms. Swarm and Evolutionary Computation 77, 101248. URL: https://doi.org/10.1016/j.swevo.2023.101248, doi:10.1016/j.swevo.2023.101248.

[19] Machado, M.C., Bellemare, M.G., Talvitie, E., Veness, J., Hausknecht, M., Bowling, M., 2018. Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents. Journal of Artificial Intelligence Research 61, 523–562. URL: https://doi.org/10.1613/jair.5699, doi:10.1613/jair.5699.

[20] Martinelli, D., de Oliveira, A.S., Kalempa, V.C., 2025. Bioinspired algorithm based on physarum polycephalum for the formation of decentralized mesh networks in multi-robot systems. Scientific Reports 16, 3457. URL: https://doi.org/10.1038/s41598-025-33456-y, doi:10.1038/s41598-025-33456-y.

[21] Mirjalili, S., Lewis, A., 2016. The Whale Optimization Algorithm. Advances in Engineering Software 95, 51–67. URL: https://doi.org/10.1016/j.advengsoft.2016.01.008, doi:10.1016/j.advengsoft.2016.01.008.

[22] Mirjalili, S., Mirjalili, S.M., Lewis, A., 2014. Grey Wolf Optimizer. Advances in Engineering Software 69, 46–61. URL: https://doi.org/10.1016/j.advengsoft.2013.12.007, doi:j.advengsoft.2013.12.007.

[23] Nanda, S.J., Panda, G., 2014. A survey on nature inspired metaheuristic algorithms for partitional clustering. Swarm and Evolutionary Computation 16, 1–18. URL: https://doi.org/10.1016/j.swevo.2013.11.003, doi:10.1016/j.swevo.2013.11.003.

[24] Sepas-Moghaddam, A., Arabshahi, A., Yazdani, D., Dehshibi, M.M., 2012. A novel hybrid algorithm for optimization in multimodal dynamic environments, in: 2012 12th International Conference on Hybrid Intelligent Systems (HIS), IEEE. pp. 143–148. URL: https://doi.org/10.1109/HIS.2012.6421324, doi:10.1109/HIS.2012.6421324.

[25] Shabani, A., Asgarian, B., Salido, M., Asil Gharebaghi, S., 2020. Search and rescue optimization algorithm: A new optimization method for solving constrained engineering optimization problems. Expert Systems with Applications 161, 113698. URL: https://doi.org/10.1016/j.eswa.2020.113698, doi:10.1016/j.eswa.2020.113698.

[26] Storn, R., Price, K., 1997. Differential evolution – A simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11, 341–359. URL: https://doi.org/10.1023/A:1008202821328, doi:10.1023/A:1008202821328.

[27] Tanabe, R., Fukunaga, A., 2013. Success-history based parameter adaptation for Differential Evolution, in: 2013 IEEE Congress on Evolutionary Computation, IEEE. pp. 71–78. URL: https://doi.org/10.1109/CEC.2013.6557555, doi:10.1109/CEC.

2013.6557555.

[28] Tanabe, R., Fukunaga, A.S., 2014. Improving the search performance of SHADE using linear population size reduction, in: 2014 IEEE Congress on Evolutionary Computation (CEC), IEEE. pp. 1658–1665. URL: https://doi.org/10.1109/CEC.2014.6900380, doi:10.1109/CEC.2014.6900380.

[29] Teo, J., 2005. Differential evolution with self-adaptive populations, in: Khosla, R., Howlett, R.J., Jain, L.C. (Eds.), Knowledge-Based Intelligent Information and Engineering Systems, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 1284–1290. URL: https://doi.org/10.1007/11552413_183, doi:10.1007/11552413_183.

[30] Turgut, O.E., Turgut, M.S., Kırtepe, E., 2023. A systematic review of the emerging metaheuristic algorithms on solving complex optimization problems. Neural Computing and Applications 35, 14275–14378. URL: https://doi.org/10.1007/s00521-023-08481-5, doi:10.1007/s00521-023-08481-5.

[31] Whitley, D., 1994. A genetic algorithm tutorial. Statistics and Computing 4, 65–85. URL: https://doi.org/10.1007/BF00175354, doi:10.1007/BF00175354.

[32] Wu, G., Mallipeddi, R., Suganthan, P.N., 2019. Ensemble strategies for population-based optimization algorithms – A survey. Swarm and Evolutionary Computation 44, 695–711. URL: https://doi.org/10.1016/j.swevo.2018.08.015, doi:10.1016/j.swevo.2018.08.015.

[33] Zhang, J., Sanderson, A.C., 2009. JADE: Adaptive Differential Evolution With Optional External Archive. IEEE Transactions on Evolutionary Computation 13, 945–958. URL: https://doi.org/10.1109/TEVC.2009.2014613, doi:10.1109/TEVC.2009.2014613.

[34] Zheng, B., Chen, Y., Wang, C., Heidari, A.A., Liu, L., Chen, H., 2024. The moss growth optimization (mgo): concepts and performance. Journal of Computational Design and Engineering 11, 184–221. URL: https://doi.org/10.1093/jcde/qwae080, doi:10.1093/jcde/qwae080.