

Article

Search-Trajectory Patterns as Feedback for Adaptive Metaheuristic Configuration Control

José Villamayor ¹, Lucas Erazo ¹, Ricardo Soto ¹, José García ² and Emanuel Vega ^{1,*}

¹ Escuela de Ingeniería Informática, Pontificia Universidad Católica de Valparaíso, Valparaíso 2362807, Chile; jose.villamayor@pucv.cl (J.V.); lucas.erazo@pucv.cl (L.E.); ricardo.soto@pucv.cl (R.S.)

² Escuela de Ingeniería de Construcción y Transporte, Pontificia Universidad Católica de Valparaíso, Valparaíso 2362804, Chile

* Correspondence: emanuel.vega@pucv.cl

Abstract

Adaptive decisions are often based on algorithm-specific feedback or simple indicators that provide limited information about the evolving search dynamics. This work investigates explicit pattern-based analysis of the search trajectory as a source of feedback for adaptive configuration control. A trajectory-driven framework is proposed in which Derivative Dynamic Time Warping (DDTW) analyzes the recent best-so-far fitness trajectory against reference patterns representing sustained progress and lack of progress. The resulting information regulates switching between predefined exploration- and exploitation-oriented parameter configurations. Two control strategies are introduced: Binary-Simple, which reacts directly to trajectory evidence associated with a plateau, and Binary-Complex, which incorporates multiple search conditions, temporal persistence, and asymmetric switching. The framework is evaluated on 27 Multidimensional Knapsack Problem instances using BPSO, GA, BGWO, and BDE, with 31 independent runs per condition. Results show that no fixed configuration consistently dominates across metaheuristics and instances, while the effects of trajectory-driven adaptation depend on the underlying algorithm. Across 108 metaheuristic–instance conditions, Binary-Complex achieves the best mean rank (2.13). The Friedman test detects an overall difference among strategies ($p = 7.73 \times 10^{-5}$); Bonferroni-Dunn post-hoc comparisons favor Binary-Complex over both fixed configurations, but do not establish a difference from Binary-Simple. Moreover, a common trajectory-analysis parameterization produces different switching behaviors across the evaluated metaheuristics. These findings support explicit search-trajectory pattern analysis as a viable feedback mechanism for adaptive configuration control across different population-based metaheuristics.

Keywords: search trajectory analysis; adaptive configuration control; adaptive parameter control; metaheuristics; derivative dynamic time warping



Academic Editor: Renato Bruni

Received: 31 August 2026

Revised: 5 October 2026

Accepted: 8 October 2026

Published: 13 October 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

MSC: 90C59

1. Introduction

Population-based metaheuristics provide flexible search mechanisms for complex optimization problems by iteratively combining diversification of the search space with intensification around promising solutions. Achieving an effective balance between these behaviors is critical, since excessive exploration may slow convergence, whereas excessive exploitation may lead to premature convergence and loss of search diversity [1–3].

More importantly, the desirable balance is not necessarily constant during an optimization run, where different stages or conditions of the search may benefit from different algorithmic behaviors [3,4]. This dynamic nature of the search has motivated adaptive mechanisms capable of exploiting information generated during optimization to modify search behavior online [5].

Adaptive mechanisms have progressively incorporated richer sources of search information, including successful parameter histories [6], explicit evolutionary-state estimation [7], stagnation-related indicators [8], and learning-based feedback [9]. These developments highlight the relevance of characterizing the ongoing search before deciding how an algorithm should adapt. Nevertheless, many existing mechanisms remain closely coupled to particular metaheuristics, their internal parameters, or specific search indicators, while explicit pattern-based analysis of the search trajectory remains comparatively less explored as a cross-algorithm source of feedback. These observations motivate investigating whether patterns in the temporal evolution of the search trajectory can provide a source of feedback for regulating exploration- and exploitation-oriented configurations across different metaheuristics.

Accordingly, this work proposes a trajectory-driven adaptive configuration-control framework in which Derivative Dynamic Time Warping (DDTW) is employed to analyze patterns in the recent best-so-far fitness trajectory. Rather than independently adapting individual control parameters, each metaheuristic operates between two predefined parameter configurations associated with exploration- and exploitation-oriented search behaviors. Information derived from the DDTW-based trajectory analysis is then used to regulate the transition between these configurations. In this way, DDTW acts as a search-monitoring mechanism within a broader adaptive controller, allowing configuration changes to respond to patterns observed in the evolving fitness trajectory.

The proposed framework is evaluated on the Multidimensional Knapsack Problem (MKP) using four population-based metaheuristics from different algorithmic families: Binary Particle Swarm Optimization (BPSO), Genetic Algorithm (GA), Binary Grey Wolf Optimizer (BGWO), and Binary Differential Evolution (BDE). For each metaheuristic, two fixed-configuration baselines are considered, maintaining either exploration- or exploitation-oriented parameters throughout the search. Two DDTW-driven adaptive variants are then evaluated under the same experimental conditions. Both employ discrete configuration switching but differ in controller complexity: Binary-Simple reacts directly to trajectory evidence associated with a plateau, whereas Binary-Complex combines multiple trajectory and progress conditions with temporal persistence and asymmetric switching. This design enables the effect of trajectory-driven adaptation to be examined across different metaheuristic paradigms while also assessing whether increased controller complexity translates into improved optimization performance.

This study focuses on using search-trajectory pattern comparisons as feedback for switching between predefined configurations. This describes the scope of the proposed controller and does not establish an advantage over adaptive parameter-control or hyper-heuristic methods. The main contributions of this work are summarized as follows:

- A trajectory-driven adaptive configuration-control framework that analyzes patterns in the fitness trajectory and uses the resulting information as feedback to regulate exploration and exploitation during the search.
- A DDTW-based realization of the proposed framework comprising discrete binary adaptation strategies with varying levels of structural complexity, allowing an assessment of different configuration-switching mechanisms.

- A cross-metaheuristic experimental assessment involving BPSO, GA, BGWO, and BDE, designed to evaluate the applicability of the same trajectory-driven control principle across different discrete population-based optimization paradigms.
- A controlled comparison against fixed exploration- and exploitation-oriented configurations to quantify the effect of trajectory-driven adaptation and determine whether increased controller complexity translates into improved optimization performance.

The remainder of this paper is organized as follows. Section 4 presents the controller, and Section 5 describes the experimental design. Section 6 reports and discusses the findings. Finally, Section 7 concludes the paper and outlines its limitations and future directions.

2. Related Work

The performance of evolutionary and swarm-based metaheuristics is strongly influenced by their parameter configuration, since parameters regulate key components of the search process and may substantially affect the resulting optimization behavior. Thus, parameter setting has been recognized as a fundamental issue in evolutionary computation [10,11]. In the literature, a well-known distinction is made between parameter tuning, where suitable parameter values are determined before the optimization process, and parameter control, where parameter values are modified during execution [12,13].

It is relevant to illustrate that, while parameter tuning seeks a suitable configuration before the search begins, parameter control allows the algorithm to modify its behavior as the optimization process evolves. This distinction is particularly relevant because appropriate parameter values may depend not only on the problem or problem instance, but also on the current stage of the search. In this context, adaptive parameter control exploits feedback obtained from the ongoing optimization process to guide parameter changes [5]. Different forms of search feedback and adaptation mechanisms have been investigated in the literature [14,15], making both the information used to characterize the search and the mechanism that translates such information into adaptation decisions central aspects of adaptive parameter control.

Following this perspective, adaptive parameter-control approaches differ substantially in the type of search information used to guide their adaptation decisions. Early contributions already demonstrated the potential of exploiting information generated during the search for online adaptation. For instance, JADE uses information associated with successful parameter values to adapt the scaling factor F and crossover rate CR [6]. This principle was subsequently extended in SHADE, which incorporates a historical memory of successful parameter settings to guide the generation of new values of F and CR [16]. Beyond information derived from successful search outcomes, early adaptive approaches also explored a more explicit characterization of the current search state using information from the population itself. A representative example is the adaptive PSO proposed in [7], which estimates the evolutionary state from population-distribution and fitness information, distinguishing among exploration, exploitation, convergence, and jumping-out states, and adapts the algorithm accordingly. Collectively, these early approaches demonstrate how different forms of information generated during the search can be exploited to characterize search behavior and guide online adaptation.

More recent studies have expanded the range of information used to drive online adaptation. In [17], the authors explored the spatial distribution of candidate solutions as an alternative feedback mechanism for adaptive parameter control, showing that information beyond conventional fitness-based feedback can be exploited during the search. Learning-based mechanisms have also been investigated; for instance, Olivares et al. employed reinforcement learning to dynamically adapt PSO parameters during execution [18]. Other

approaches have focused on detecting undesirable search conditions, using covariance-based population information [19] or stagnation indicators [8] to characterize search behavior and adapt algorithm parameters accordingly. More recently, adaptive parameter control has also been investigated in parallel island-based metaheuristics, where search progress and performance feedback are used to dynamically adjust algorithm parameters [20].

Collectively, these studies demonstrate the value of search-generated information for online adaptation. However, existing mechanisms are commonly tailored to specific metaheuristics and control parameters, while explicit pattern-based analysis of the search trajectory remains comparatively less explored as a cross-algorithm source of feedback. This motivates the investigation of trajectory-driven mechanisms for adaptive configuration control across different metaheuristics.

3. Background

This section introduces the main elements required to describe the proposed framework. First, the Multidimensional Knapsack Problem (MKP) is presented as the optimization problem used for evaluation. Next, the four population-based metaheuristics considered in this study are briefly characterized. Finally, Dynamic Time Warping (DTW) and Derivative Dynamic Time Warping (DDTW) are introduced as the temporal-analysis techniques underlying the trajectory-pattern analysis developed in the proposed approach.

3.1. Multidimensional Knapsack Problem

The Multidimensional Knapsack Problem (MKP) is a classical NP-hard combinatorial optimization problem in which a subset of items must be selected to maximize the total profit while satisfying multiple resource constraints. Given n items, each item j is associated with a profit p_j , while w_{ij} denotes the consumption of resource i by item j . Each resource i has a maximum available capacity c_i . The binary decision variable x_j indicates whether item j is selected. The problem can be formulated as

$$\begin{aligned} \max_{\mathbf{x}} \quad & \sum_{j=1}^n p_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^n w_{ij} x_j \leq c_i, \quad i = 1, \dots, m, \\ & x_j \in \{0, 1\}, \quad j = 1, \dots, n. \end{aligned} \quad (1)$$

The MKP has been extensively studied in combinatorial optimization and metaheuristics, and several standardized benchmark sets are available for experimental evaluation. In this work, the benchmark instances introduced by Chu and Beasley are considered [21].

3.2. Population-Based Metaheuristics

Four population-based metaheuristics from different algorithmic families are considered in this study: Binary Particle Swarm Optimization (BPSO), Genetic Algorithm (GA), Binary Grey Wolf Optimizer (BGWO), and Binary Differential Evolution (BDE). Their inclusion provides different population-update mechanisms and search dynamics on which the proposed trajectory-driven control principle can be evaluated. Table 1 summarizes their main characteristics.

Although these algorithms differ in their underlying search mechanisms, each one contains internal parameters that influence the balance between diversification and intensification. The specific parameters and exploration- and exploitation-oriented configurations used in this work are defined later as part of the proposed approach.

Table 1. Population-based metaheuristics considered in this study.

MH	Family	Main Search Mechanism	Binary Representation
BPSO	Swarm intelligence	Particle velocities are updated according to inertia, individual experience, and global swarm information [22].	Sigmoid transfer function
GA	Evolutionary computation	Candidate solutions evolve through selection, crossover, and mutation operators [23,24].	Native binary representation
BGWO	Swarm intelligence	Candidate solutions update their positions according to the hierarchical guidance provided by the α , β , and δ wolves [25,26].	Transfer-function-based binarization
BDE	Evolutionary computation	New candidate solutions are generated through differential mutation followed by crossover and selection [27,28].	Transfer-function-based binarization

3.3. Dynamic Time Warping and Derivative Dynamic Time Warping

Dynamic Time Warping (DTW) is a dynamic-programming technique originally developed for comparing temporal sequences that may exhibit nonlinear displacements along the time axis [29]. Unlike a direct point-to-point distance, DTW searches for an alignment between two sequences while allowing elements from one sequence to be matched with multiple elements from the other.

Let

$$A = (a_1, a_2, \dots, a_n) \quad (2)$$

and

$$B = (b_1, b_2, \dots, b_m) \quad (3)$$

be two temporal sequences of lengths n and m , respectively. DTW constructs an accumulated-cost matrix $D \in \mathbb{R}^{(n+1) \times (m+1)}$, where $D[i, j]$ represents the minimum cost required to align the prefixes $(a_1, \dots, a_i)$ and $(b_1, \dots, b_j)$. Using a local distance $d(a_i, b_j)$, the recurrence relation is

$$D[i, j] = d(a_i, b_j) + \min\{D[i-1, j], D[i, j-1], D[i-1, j-1]\}. \quad (4)$$

The boundary conditions are

$$D[0, 0] = 0, \quad D[i, 0] = \infty, \quad D[0, j] = \infty, \quad (5)$$

for $i > 0$ and $j > 0$. All other cells are initially set to ∞ . The evaluated implementation uses the local absolute-difference cost, not a squared Euclidean cost:

$$d(a_i, b_j) = |a_i - b_j|. \quad (6)$$

The returned dissimilarity is the accumulated cost $D[n, m]$. No square root or division by sequence length or warping-path length is applied. Thus, the reported DDTW values are raw accumulated costs, not normalized distances.

A limitation of conventional DTW is that differences in the absolute amplitude of two sequences may induce undesirable temporal warpings. To reduce this effect, Keogh and Pazzani proposed Derivative Dynamic Time Warping (DDTW), which performs the alignment on estimates of the local derivatives rather than directly on the original sequence values [30]. Hence,

$$\text{DDTW}(A, B) = \text{DTW}(A', B'), \quad (7)$$

where the evaluated derivative representation is the first difference

$$a'_i = \begin{cases} 0, & i = 1, \\ a_i - a_{i-1}, & 2 \leq i \leq n, \end{cases} \quad (8)$$

and B' is defined identically. This preserves the sequence length and sets the first derivative sample to zero; the final sample uses the backward difference. This implementation follows the derivative-comparison principle of DDTW, but uses first differences rather than the original smoothed derivative estimator. No amplitude normalization or fitness rescaling is applied before differentiation.

The use of derivatives shifts the comparison from absolute fitness levels toward local variations in the temporal evolution of the sequences. In particular, additive vertical offsets are removed under differentiation, since for a shifted sequence

$$\tilde{A} = A + c, \quad (9)$$

its derivative satisfies

$$\tilde{A}' = A'. \quad (10)$$

This property is particularly relevant when the objective is to compare the shape of search trajectories rather than their absolute objective-function values. To restrict excessive temporal deformation, DTW can additionally be constrained using a Sakoe–Chiba band [29]. For a half-width w , only matrix cells satisfying

$$|i - j| \leq w \quad (11)$$

are considered during alignment. Classical DTW requires $O(nm)$ time for sequences of lengths n and m . For sequences of comparable length n , this becomes $O(n^2)$, whereas the Sakoe–Chiba restriction reduces the number of recurrence evaluations to approximately $O(nw)$. The effective band is $\max(w, |n - m|)$, so the endpoint remains reachable for unequal lengths; cells outside this band retain infinite cost. The evaluated Python implementation still allocates and initializes the full cost matrix, requiring $O(nm)$ memory and initialization work. Hence the band restriction does not make the complete implementation linear in n .

The proposed approach uses DDTW to compare recent segments of the fitness trajectory with predefined reference patterns. The construction of these patterns, the trajectory window, the resulting feedback signal, and its use for adaptive configuration control are described in Section 4.

4. Proposed Approach

4.1. Trajectory-Driven Adaptive Configuration-Control Framework

The proposed framework couples three main components: (i) a population-based metaheuristic responsible for solving the optimization problem, (ii) a trajectory-analysis module that continuously characterizes the recent evolution of the search, and (iii) an adaptive controller that translates the resulting trajectory information into changes in the operating configuration of the metaheuristic. Figure 1 illustrates the resulting closed-loop architecture.

At iteration t , the metaheuristic evaluates a population of candidate solutions and records the best-so-far fitness value, denoted by f_t^* . The trajectory-analysis module maintains a sliding window containing the most recent W observations of this convergence trajectory. During an initial warm-up period, before a complete window is available, the metaheuristic operates under its initial configuration. Once W observations have been col-

lected, the trajectory window is compared with predefined reference patterns representing sustained progress and lack of progress. Both adaptive strategies start in exploration. During warm-up, the monitor supplies zero distance placeholders and marks its output as not ready: Binary-Simple retains exploration because of its plateau condition is true, whereas Binary-Complex explicitly preserves its current state. No DDTW comparison is made before the window is complete. The resulting dissimilarity measures are subsequently supplied to the adaptive controller, which determines the configuration to be used by the metaheuristic. Specifically, the mode s_t generates the fitness observation at iteration t , and the controller decision computed from that observation determines s_{t+1} . All switching rules below use this observation-then-decision convention.

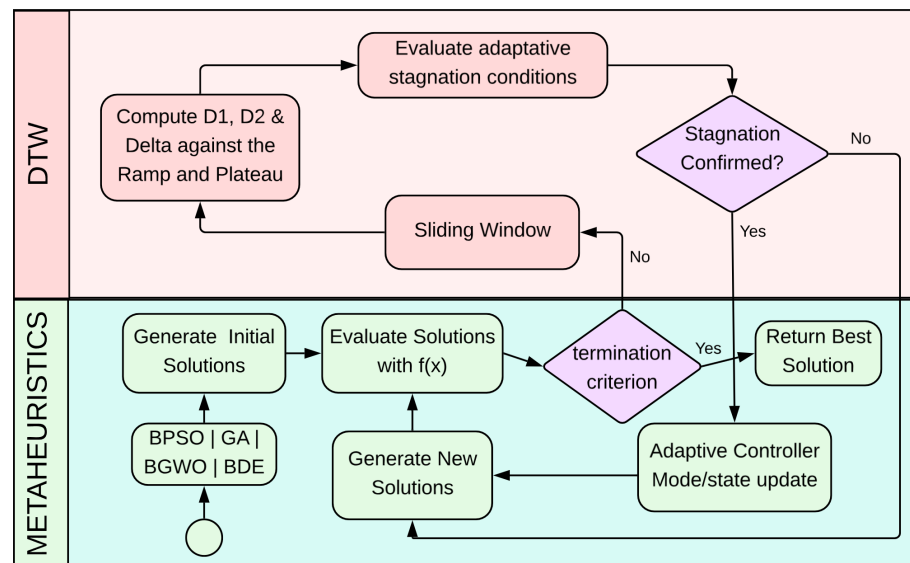


Figure 1. Closed-loop architecture of the proposed trajectory-driven adaptive configuration-control framework.

The architecture is deliberately modular. The trajectory-analysis module is independent of the particular update equations of the underlying metaheuristic, whereas the adaptive controller operates on a common configuration interface rather than directly on algorithm-specific search operators. Consequently, the same trajectory information can be processed by different decision strategies and applied to different metaheuristics by mapping the resulting control signal to their respective internal parameters. This separation enables the trajectory-analysis mechanism, decision logic, and underlying optimization algorithm to be studied under a common experimental framework.

4.2. DDTW-Based Trajectory-Pattern Analysis

The trajectory-analysis module characterizes the recent search dynamics by comparing the observed convergence trajectory with predefined geometric reference patterns. Rather than representing lack of progress exclusively through the number of consecutive iterations without improvement, the proposed mechanism evaluates the shape of the recent fitness trajectory using Derivative Dynamic Time Warping (DDTW). The resulting distances quantify whether the observed search dynamics resemble sustained progress or a plateau and constitute the feedback information supplied to the adaptive controller.

4.2.1. Reference Trajectory Patterns

Let

$$X_t = (f_{t-W+1}^*, f_{t-W+2}^*, \dots, f_t^*) \quad (12)$$

denote the sliding window containing the W most recent best-so-far fitness values. Two synthetic reference patterns are constructed to represent contrasting search behaviors.

The first pattern, denoted by R_t , represents sustained progress through a linear trajectory with reference slope $s_{\min} > 0$:

$$r_i = f_{t-W+1}^* + s_{\min}(i-1), \quad i = 1, \dots, W. \quad (13)$$

The second pattern, denoted by C_t , represents the absence of progress through a constant trajectory:

$$c_i = f_{t-W+1}^*, \quad i = 1, \dots, W. \quad (14)$$

Thus, R_t provides a reference for sustained improvement, whereas C_t represents a fully stagnant trajectory. Since the comparison is performed on derivative representations through DDTW, the analysis emphasizes the local evolution of the trajectory rather than its absolute fitness level.

These patterns are operational endpoints for low and sustained progress, not a complete model of the population dynamics of all four algorithms. The fixed slope $s_{\min} = 2.0$ represents two objective-function units per iteration. It is not estimated from the current window or fitted separately for each algorithm or instance. Choosing the same reference makes the comparison procedure common across algorithms, but does not establish that this numerical slope is optimal or scale-invariant. Additive fitness shifts are removed by first differences; multiplicative fitness rescaling is not. The percentiles adapt to each run's distance history, so absolute distances should not be compared directly across different objective scales. The OAT study in Section 6.5 does not vary this slope.

4.2.2. Trajectory Dissimilarity Measures

The dissimilarity between the observed trajectory and each reference pattern is quantified using DDTW with a Sakoe–Chiba constraint of half-width w_{SC} :

$$D_R(t) = \text{DDTW}(X_t, R_t), \quad D_C(t) = \text{DDTW}(X_t, C_t). \quad (15)$$

A relative trajectory indicator is additionally defined as

$$\Delta_t = D_R(t) - D_C(t). \quad (16)$$

Here, $D_R(t)$ measures the dissimilarity between the recent trajectory and the sustained-progress reference, whereas $D_C(t)$ measures its dissimilarity from the constant reference. Consequently, Δ_t provides a relative comparison between both patterns: negative values indicate that the observed trajectory is closer to the progress reference, whereas positive values indicate greater similarity to the plateau reference.

For the constant reference, the first-difference sequence is identically zero. Every admissible DTW path visits each observed derivative at least once, and the diagonal path attains the sum of their absolute values. Writing $x_i = f_{t-W+i}^*$ for $i = 1, \dots, W$, the nondecreasing best-so-far window used here gives

$$D_C(t) = \sum_{i=2}^W |x_i - x_{i-1}| = x_W - x_1. \quad (17)$$

This identity holds for any band admitting the diagonal. Binary-Simple's plateau score therefore measures cumulative progress within the window; it does not uniquely distinguish detailed trajectory shapes with the same endpoint gain. Binary-Complex additionally uses the ramp comparison and temporal conditions. This distinction limits

the interpretation of the plateau score and explains why changing the band alone cannot change it.

4.2.3. History-Based Adaptive Thresholds

Because the magnitude and distribution of the trajectory distances may vary during the optimization process, the decision thresholds are not defined as fixed absolute values. Instead, they are dynamically estimated from the historical distributions of the trajectory metrics.

Let $\mathcal{H}_R^{(t)}$, $\mathcal{H}_C^{(t)}$, and $\mathcal{H}_\Delta^{(t)}$ denote the accumulated histories of D_R , D_C , and Δ available at iteration t , respectively. The adaptive thresholds are defined as

$$\theta_C(t) = \mathcal{P}_{p_{\text{low}}}(\mathcal{H}_C^{(t)}), \quad (18)$$

$$\theta_R(t) = \mathcal{P}_{p_{\text{high}}}(\mathcal{H}_R^{(t)}), \quad (19)$$

and

$$\theta_\Delta(t) = \mathcal{P}_{p_{\text{high}}}(\mathcal{H}_\Delta^{(t)}), \quad (20)$$

where $\mathcal{P}_p(\cdot)$ denotes the p -th percentile of the corresponding historical distribution. The lower percentile p_{low} identifies unusually small distances to the plateau reference, whereas the upper percentile p_{high} identifies unusually large deviations from the sustained-progress reference or correspondingly large values of the relative trajectory indicator.

This history-based formulation allows the decision criteria to adapt to the distance distributions observed during each optimization run, reducing their dependence on manually specified absolute distance thresholds after the initial history-building phase. The current observation is appended before computing percentiles, and all complete-window observations remain in each history; there is no rolling truncation of these histories. Percentiles use linear interpolation between sorted samples, as in `numpy.percentile` with its default method. If fewer than ten complete-window observations have been accumulated, the implementation uses

$$\theta_C = 0.1W, \quad \theta_R = 0.5W, \quad \theta_\Delta = 0.3W. \quad (21)$$

Thus, with $W = 200$, the first complete window is observed at iteration 200 and history-based percentiles begin at iteration 209. The same fixed thresholds are returned as placeholders during warm-up, together with a not-ready flag. The consecutive-no-improvement counter is zero for the first post-iteration observation; subsequently it increments on a tie or decrease and resets on a strict improvement. The persistence counter increments only when all three stagnation conditions hold and otherwise resets to zero.

4.3. Adaptive Configuration-Control Strategies

The trajectory information described above is translated into changes in the search configuration through two binary adaptive control strategies. Both strategies operate on the same trajectory information and configuration interface but differ in the complexity of the decision mechanism used to trigger transitions between exploration and exploitation.

For binary control, let

$$s_t \in \{0, 1\} \quad (22)$$

denote the search mode at iteration t , where $s_t = 0$ selects the exploitation-oriented configuration, and $s_t = 1$ selects the exploration-oriented configuration. The corresponding parameter vector can be expressed as

$$\theta_t = \begin{cases} \theta^{\text{exploit}}, & s_t = 0, \\ \theta^{\text{explore}}, & s_t = 1, \end{cases} \quad (23)$$

where θ^{exploit} and θ^{explore} denote the two predefined algorithm-specific parameter configurations. Their numerical values for BPSO, GA, BGWO, and BDE are reported in the Experimental Setup.

4.3.1. Binary-Simple Strategy

Binary-Simple constitutes the lowest-complexity trajectory-driven controller. It uses only the distance to the plateau reference and does not require temporal persistence of the detected condition. The decision rule is

$$s_{t+1} = \begin{cases} 1, & D_C(t) \leq \theta_C(t), \\ 0, & \text{otherwise.} \end{cases} \quad (24)$$

Therefore, when the recent trajectory becomes unusually similar to the plateau reference with respect to its historical behavior, the controller selects the exploration-oriented configuration. Otherwise, the exploitation-oriented configuration is used.

4.3.2. Binary-Complex Strategy

Binary-Complex extends the previous controller by combining multiple trajectory and progress conditions with temporal persistence and asymmetric switching. Three conditions are evaluated:

$$C_{\text{noimp}}(t) : \ell_{\text{noimp}}(t) \geq \pi_{\text{max}}, \quad (25)$$

$$C_{\text{plateau}}(t) : D_C(t) \leq \theta_C(t), \quad (26)$$

$$C_{\text{progress}}(t) : [D_R(t) \geq \theta_R(t)] \vee [\Delta_t \geq \theta_\Delta(t)]. \quad (27)$$

Here, $\ell_{\text{noimp}}(t)$ denotes the number of consecutive iterations without improvement in the best-so-far fitness, and π_{max} defines the minimum number of such iterations required to provide explicit evidence of lack of progress. The second condition identifies trajectories unusually close to the plateau reference, whereas the third captures trajectories that are unusually distant from the sustained-progress pattern, either directly through $D_R(t)$ or relatively through Δ_t .

Let

$$C_{\text{stag}}(t) = C_{\text{noimp}}(t) \wedge C_{\text{plateau}}(t) \wedge C_{\text{progress}}(t) \quad (28)$$

denote the composite stagnation condition. A transition from exploitation to exploration is performed only when $C_{\text{stag}}(t)$ remains satisfied for τ_{pat} consecutive iterations. In contrast, a strict improvement in the best-so-far fitness immediately returns the algorithm to the exploitation-oriented configuration:

$$s_{t+1} = \begin{cases} 1, & s_t = 0 \wedge C_{\text{stag}}(k) = 1 \quad \forall k \in \{t - \tau_{\text{pat}} + 1, \dots, t\}, \\ 0, & s_t = 1 \wedge f_t^* > f_{t-1}^*, \\ s_t, & \text{otherwise.} \end{cases} \quad (29)$$

This asymmetric hysteresis requires sustained evidence of stagnation before activating exploration, thereby reducing sensitivity to transient trajectory fluctuations. Conversely, once the exploratory configuration produces a strict improvement in the best-so-far solution, the controller immediately returns to exploitation for the next iteration so that the newly discovered region can be intensified. These transitions are enabled only after the monitor becomes ready; during warm-up Binary-Complex retains the initial exploration mode. The persistence counter is updated by the monitor on every ready observation, independently of the controller mode.

5. Experimental Setup

5.1. Environment

The computational experiments were conducted within a Python 3.11.0 execution environment, leveraging the high-throughput capabilities of the Océano High-Performance Computing (HPC) Cluster at the Pontificia Universidad Católica de Valparaíso (PUCV), Valparaíso, Chile. Workloads were orchestrated via the Slurm Workload Manager 23.11.4 across high-density Dell PowerEdge R640 compute nodes (equipped with dual Intel® Xeon® Scalable processors and ECC DDR4 RAM). The equipment was provided on loan by the Océano HPC Cluster at PUCV, Valparaíso, Chile. Vectorized routines were accelerated using NumPy 2.1.3, and preliminary script prototyping was carried out on a dedicated multi-core workstation.

5.2. Benchmark Instances and Metaheuristic Configurations

The proposed framework is evaluated using benchmark instances from the Chu–Beasley Multidimensional Knapsack Problem (MKP) test suite [21], available through the OR-Library. Nine benchmark groups are considered, corresponding to the combinations of three numbers of constraints ($m \in \{5, 10, 30\}$) and three problem dimensions ($n \in \{100, 250, 500\}$). Three instances are selected from each group using the fixed, zero-based indices 0, 15, and 29, yielding 27 evaluation instances. Index 0 retains the original benchmark cases for backward comparison; the two additional indices extend coverage within each group. This deterministic selection is not a random or exhaustive sample of the 30 instances available in each group. Throughout the results, `mknapcbg[i]` denotes index i from benchmark group g . Table 2 summarizes the dimensions and the known optimal objective values for all selected cases.

Table 2. The nine MKP benchmark groups and the three selected instances per group (zero-based indices 0, 15, and 29): 27 instances in total.

Group	m	n	f^* (idx 0)	f^* (idx 15)	f^* (idx 29)
mknapcb1	5	100	24,381	42,927	59,965
mknapcb2	5	250	59,312	110,256	154,662
mknapcb3	5	500	120,130	220,514	299,904
mknapcb4	10	100	23,064	42,995	60,633
mknapcb5	10	250	59,187	110,841	149,704
mknapcb6	10	500	117,726	215,013	307,014
mknapcb7	30	100	21,946	41,058	60,603
mknapcb8	30	250	56,693	107,246	149,572
mknapcb9	30	500	115,868	215,762	300,460

Candidate solutions that violate one or more resource constraints are restored to feasibility using a greedy repair procedure. Infeasible solutions are first repaired by iteratively removing low-density items until all constraints are satisfied. A subsequent insertion phase considers high-density items and adds them whenever their inclusion preserves feasibility.

The same repair procedure is applied to all metaheuristics and adaptive variants to avoid introducing algorithm-dependent differences in constraint handling.

Metaheuristic Configurations

All four metaheuristics use a population size of $N = 20$ and a computational budget of $T = 2000$ iterations. Each experimental condition is evaluated over $R = 31$ independent runs. These settings are kept fixed across algorithms, instances, and control strategies.

Four experimental variants are considered for each metaheuristic: Vanilla-Exploitation, Vanilla-Exploration, Binary-Simple, and Binary-Complex. The two Vanilla variants maintain a fixed exploitation or exploration-oriented parameter configuration throughout the entire run, whereas Binary-Simple and Binary-Complex dynamically switch between the same two configurations according to the trajectory-driven control mechanisms described in Section 4.3. For BPSO, BGWO, and BDE, continuous internal quantities are mapped to binary decisions using a sigmoid transfer function. Table 3 reports the parameter values defining the two configurations for each metaheuristic.

Table 3. Exploration- and exploitation-oriented parameter configurations used for each metaheuristic.

MH	Parameter	Exploitation	Exploration
BPSO	w	0.729	0.9
	c_1	1.49445	2.5
	c_2	1.49445	0.5
GA	p_c	0.9	0.6
	p_m	0.01	0.15
BGWO	a	0.5	2.0
BDE	F	0.5	0.9
	CR	0.9	0.3

The parameter configurations were defined using established parameter settings and search-behavior principles reported for the corresponding metaheuristics [24,25,27,31]. The exploitation-oriented profiles favor stronger convergence toward promising regions, whereas the exploration-oriented profiles modify the corresponding parameters to promote greater diversification. Specifically, the BPSO exploration profile increases the inertia and cognitive contribution while reducing the social component; the GA profile increases the mutation probability while reducing the crossover probability; the BGWO profile uses a larger value of the exploration-related coefficient a ; and the BDE profile increases the differential scaling factor while reducing the crossover rate. These configurations are kept unchanged across all benchmark instances. The cited studies motivate the parameter roles and the direction of these changes; they do not certify the numerical profiles as optimal for MKP. In the evaluated implementation, these profiles are fixed constants rather than outputs of a per-instance fitting procedure. No systematic search over profile values or a separate training/validation phase is reported here. Accordingly, the experiment evaluates switching between two prescribed profiles, not the optimality of their selection.

5.3. DDTW Trajectory-Analysis and Controller Parameters

The DDTW-based trajectory-analysis module and both adaptive controllers use a common parameterization throughout the complete experimental evaluation. No instance- or metaheuristic-specific adjustment of these parameters is performed. Table 4 summarizes the corresponding values.

The percentile values $p_{\text{low}} = 40$ and $p_{\text{high}} = 60$ are used to dynamically derive the trajectory thresholds from the historical distributions accumulated during each run. Binary-

Simple uses only the plateau-distance condition $D_C(t) \leq \theta_C(t)$, whereas Binary-Complex combines the three conditions defined in Section 4.3 together with the no-improvement threshold $\pi_{\max}$ and persistence parameter τ_{pat} .

The same values of W , w_{SC} , $s_{\min}$, p_{low} , and p_{high} are used by both adaptive strategies. Consequently, differences between Binary-Simple and Binary-Complex arise from their decision logic rather than from different parameterizations of the underlying trajectory-analysis module.

Table 4. Parameters of the DDTW-based trajectory-analysis module and adaptive controllers.

Parameter	Value	Description
W	200	Sliding-window length
w_{SC}	2	Sakoe–Chiba band half-width
$s_{\min}$	2.0	Reference slope for sustained progress
p_{low}	40	Percentile defining θ_C
p_{high}	60	Percentile defining θ_R and θ_Δ
$\pi_{\max}$	5	Minimum consecutive iterations without improvement
τ_{pat}	3	Persistence required for stagnation confirmation

5.4. Experimental Protocol

Each experimental run corresponds to the application of one metaheuristic under one configuration-control strategy to a given MKP instance. The same population size, iteration budget, repair mechanism, and trajectory-analysis parameters are maintained across the corresponding experimental conditions. The expanded campaign contains $27 \times 4 \times 4 = 432$ experimental conditions and $432 \times 31 = 13,392$ runs, with 31 independent replicates per condition. All 432 serialized fitness arrays contain 31 finite values; the solution-quality tables were checked against these arrays before incorporating the expanded results. This campaign is separate from the one-factor-at-a-time sensitivity experiment in Section 6.5. Algorithm 1 summarizes the procedure for a single independent run.

Algorithm 1 Single run with explicit configuration-switching rules.

Require: Instance $\mathcal{I}$, algorithm $\mathcal{A}$, strategy $\mathcal{C}$, profiles Π_0, Π_1 , seed q , population N , budget T , monitor settings
Ensure: Best solution and in-memory fitness, diagnostic, and applied-mode histories

- 1: Initialize `numpy.random.default_rng` with seed q
- 2: Generate, repair, and evaluate the initial population of size N
- 3: Initialize the incumbent $\mathbf{x}^*$ and empty post-iteration histories
- 4: Initialize monitor history, no-improvement count $\ell = 0$, persistence $g = 0$
- 5: Set $s_1 = 0$ for Vanilla-Exploitation; otherwise set $s_1 = 1$
- 6: Apply profile Π_{s_1} after population initialization
- 7: **for** $t = 1, \dots, T$ **do**
- 8: Generate, repair, and evaluate candidates using profile Π_{s_t}
- 9: Update $\mathbf{x}^*$ and obtain best-so-far fitness f_t^*
- 10: Record f_t^* and the mode s_t actually used in this iteration
- 11: Set $s_{t+1} \leftarrow s_t$
- 12: **if** $\mathcal{C}$ is adaptive **then**
- 13: Obtain monitor output $\mathbf{z}_t$ using Algorithm 2
- 14: **if** $\mathcal{C}$ is Binary-Simple **then**
- 15: $s_{t+1} \leftarrow [D_C(t) \leq \theta_C(t)]$ ▷ Includes warm-up placeholders
- 16: **else if** the monitor is ready **then**
- 17: **if** $s_t = 0$ and $g \geq \tau_{\text{pat}}$ **then**
- 18: $s_{t+1} \leftarrow 1$
- 19: **else if** $s_t = 1$ and $\ell = 0$ **then**
- 20: $s_{t+1} \leftarrow 0$
- 21: **end if**
- 22: **end if**
- 23: Apply profile $\Pi_{s_{t+1}}$ for the next iteration
- 24: Record diagnostics and count a transition only if $s_t = 0, s_{t+1} = 1$
- 25: **end if**
- 26: **end for**
- 27: **return** $\mathbf{x}^*$, final fitness, and the three in-memory histories

Algorithm 2 Monitor update: derivatives, thresholds, and temporal confirmation.

Require: Current observation f_t^* , history, W, w_{SC}, s_{min} , percentiles $p_{low}, p_{high}, \pi_{max}, \tau_{pat}$
Ensure: Ready flag, D_R, D_C, Δ , thresholds, ℓ, g , and fire flag

- 1: **if** the history is nonempty and f_t^* is not greater than its last value **then**
- 2: Increment ℓ and clamp f_t^* to the last best-so-far value
- 3: **else**
- 4: $\ell \leftarrow 0$
- 5: **end if**
- 6: Append f_t^* to the post-iteration best-so-far history
- 7: **if** fewer than W observations are available **then**
- 8: Return not-ready, zero distances and Δ , thresholds $(0.1W, 0.5W, 0.3W)$, $\ell, g = 0$, fire=false
- 9: **end if**
- 10: Extract the most recent W observations as X_t
- 11: Construct R_t, C_t using Equations (13) and (14)
- 12: First-difference X_t, R_t, C_t using Equation (8)
- 13: **for** each reference derivative $B' \in \{R'_t, C'_t\}$ **do**
- 14: Initialize $D[0:W, 0:W] \leftarrow \infty$ and $D[0, 0] \leftarrow 0$
- 15: **for** $i = 1, \dots, W$ **do**
- 16: **for** $j = \max(1, i - w_{SC}), \dots, \min(W, i + w_{SC})$ **do**
- 17: $D[i, j] \leftarrow |x'_i - b'_j| + \min\{D[i - 1, j], D[i, j - 1], D[i - 1, j - 1]\}$
- 18: **end for**
- 19: **end for**
- 20: Store $D[W, W]$ as the corresponding distance D_R or D_C
- 21: **end for**
- 22: $\Delta \leftarrow D_R - D_C$
- 23: Append these values to their respective complete-window histories
- 24: **if** at least ten complete-window observations are available **then**
- 25: Compute thresholds from Equations (18)–(20) ▷ Include current sample; linear percentiles
- 26: **else**
- 27: Set thresholds to $(0.1W, 0.5W, 0.3W)$
- 28: **end if**
- 29: $C \leftarrow [\ell \geq \pi_{max}] \wedge [D_C \leq \theta_C]$
- 30: $C \leftarrow C \wedge ([D_R \geq \theta_R] \vee [\Delta \geq \theta_\Delta])$
- 31: **if** C is true **then**
- 32: $g \leftarrow g + 1$
- 33: **else**
- 34: $g \leftarrow 0$
- 35: **end if**
- 36: Return ready, distances, thresholds, ℓ, g , and fire= $[g \geq \tau_{pat}]$

5.5. Implementation and Reproduction Details

For each condition, the HPC runner assigns seed $q = e + 1$ to the zero-based run index $e = 0, \dots, 30$. The same seed list $1, \dots, 31$ is used across the four strategies and four algorithms, with a fresh NumPy generator for each run. Results are sorted by run index before serialization, not by worker completion time. This defines the pairing used in local comparisons; different algorithms may consume different random draws even at the same seed. The explicit instance indices 0, 15, 29 are independent of this run-seed convention and are not randomly sampled in the reported campaign.

The runner creates full fitness, monitor, and applied-mode histories in memory. However, the saved campaign JSON files contain only final fitness, runtime, transition counts, gains, summary statistics, and configuration metadata. They do not retain the full per-iteration histories, explicit seed arrays, or a source commit hash. Consequently, aggregate tables can be regenerated from these files, whereas synchronized controller traces require rerunning the experiment and saving the in-memory histories. Exact runtime reproduction additionally depends on hardware, process allocation, and the software environment; an iteration budget is not a hardware-independent wall-clock budget. Algorithms 1 and 2 expose the decision sequence.

6. Results

This section presents the experimental results obtained for the four metaheuristics (BPSO, GA, BGWO, and BDE) across the 27 MKP benchmark instances (three selected indices per group). The analysis considers three complementary aspects: solution quality,

statistical evidence, and computational overhead. In addition, the switching behavior of the adaptive controllers is examined to provide insight into how the trajectory-driven mechanisms operate during the search. Each experimental condition comprises $R = 31$ independent runs.

The MFSS results cannot be interpreted as a direct performance comparison with the methods evaluated in this study. The MFSS experiments used ten independent runs and a time-based budget, whereas our experiments use 31 runs and a budget of 2000 iterations. In addition, the ID 0 objective or best-known values reported for four overlapping benchmark groups differ from the values listed in Table 2. Therefore, instance-level equivalence has not been established for those groups, and no statistical or superiority claim is made from the published MFSS values.

6.1. Solution Quality

Optimization performance is first assessed using the final best-so-far objective values obtained over the independent runs. For each experimental condition, the best fitness value (f_{best}), mean final fitness ($\bar{f}$), standard deviation (σ), and relative optimality gap are reported. Since the global optimum f^* is known for all benchmark instances, the mean relative optimality gap is computed as

$$\text{Gap}(\%) = \left(\frac{f^* - \bar{f}}{f^*} \right) \times 100. \quad (30)$$

Lower Gap values therefore indicate better average solution quality, with Gap = 0 corresponding to the known optimum.

To preserve direct comparison with the original nine-case evaluation, Tables 5–7 retain three representative index-0 cases: mknapcb1 [0] ($n = 100, m = 5$), mknapcb5 [0] ($n = 250, m = 10$), and mknapcb9 [0] ($n = 500, m = 30$). Table 8 summarizes the enlarged 27-instance evaluation. Complete per-instance solution-quality results for the remaining 24 cases are provided in Appendix A.

Table 5. Solution quality on mknapcb1 (inst. 0, $n = 100, m = 5, f^* = 24,381$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	24,381	24,256.58	46.29	0.51
BPSO—Vanilla Exploration	24,381	24,343.65	26.75	0.15
BPSO—Binary-Simple	24,381	24,329.74	32.18	0.21
BPSO—Binary-Complex	24,381	24,330.87	32.34	0.21
GA—Vanilla Exploitation	24,373	24,283.26	55.41	0.40
GA—Vanilla Exploration	24,381	24,309.35	31.09	0.29
GA—Binary-Simple	24,373	24,307.23	36.21	0.30
GA—Binary-Complex	24,381	24,316.90	38.86	0.26
BGWO—Vanilla Exploitation	24,259	24,197.77	33.65	0.75
BGWO—Vanilla Exploration	24,327	24,190.19	38.87	0.78
BGWO—Binary-Simple	24,327	24,181.58	52.96	0.82
BGWO—Binary-Complex	24,327	24,191.71	36.68	0.78
BDE—Vanilla Exploitation	24,330	24,234.52	79.08	0.60
BDE—Vanilla Exploration	24,381	24,322.32	30.15	0.24
BDE—Binary-Simple	24,381	24,306.39	41.38	0.31
BDE—Binary-Complex	24,337	24,288.06	27.59	0.38

The extended sample retains the algorithm-dependent pattern, but does not support uniform dominance by the more complex controller. Across all 27 instances, BDE's mean gap is 0.77% with Binary-Complex, compared with 1.51% for Vanilla-Exploitation and 0.94% for Vanilla-Exploration; Binary-Simple has a slightly lower descriptive average (0.75%). For

GA, Binary-Complex (0.52%) improves on Binary-Simple (0.86%) and Vanilla-Exploration (1.75%), while Vanilla-Exploitation remains stronger on average (0.41%). For BPSO, Vanilla-Exploration has the lowest average gap (1.64%), versus 1.72% for Binary-Complex. BGWO's average gaps are close across strategies (2.73%–2.79%). These summaries include the new indices 15 and 29 and complement, rather than replace, the individual-case results.

Table 6. Solution quality on mknpcb5 (inst. 0, $n = 250$, $m = 10$, $f^* = 59,187$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	58,285	57,676.74	207.82	2.55
BPSO—Vanilla Exploration	58,564	58,288.10	109.54	1.52
BPSO—Binary-Simple	58,527	58,224.81	148.57	1.63
BPSO—Binary-Complex	58,439	58,186.45	123.58	1.69
GA—Vanilla Exploitation	59,094	58,830.06	130.21	0.60
GA—Vanilla Exploration	58,573	58,206.61	144.26	1.66
GA—Binary-Simple	58,966	58,617.29	193.96	0.96
GA—Binary-Complex	58,947	58,690.10	152.55	0.84
BGWO—Vanilla Exploitation	58,148	57,569.81	187.46	2.73
BGWO—Vanilla Exploration	57,747	57,468.90	128.10	2.90
BGWO—Binary-Simple	58,058	57,532.39	183.26	2.80
BGWO—Binary-Complex	57,747	57,464.65	147.07	2.91
BDE—Vanilla Exploitation	58,807	58,394.58	193.71	1.34
BDE—Vanilla Exploration	58,938	58,628.68	139.05	0.94
BDE—Binary-Simple	58,971	58,671.94	156.25	0.87
BDE—Binary-Complex	58,961	58,747.71	111.18	0.74

Table 7. Solution quality on mknpcb9 (inst. 0, $n = 500$, $m = 30$, $f^* = 115,868$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	111,723	111,259.61	268.02	3.98
BPSO—Vanilla Exploration	112,766	112,194.90	248.78	3.17
BPSO—Binary-Simple	112,527	112,177.81	230.27	3.18
BPSO—Binary-Complex	112,752	112,155.48	284.79	3.20
GA—Vanilla Exploitation	115,205	114,841.97	207.64	0.89
GA—Vanilla Exploration	112,886	112,113.00	327.65	3.24
GA—Binary-Simple	114,665	113,450.29	579.15	2.09
GA—Binary-Complex	114,647	114,272.58	244.37	1.38
BGWO—Vanilla Exploitation	111,836	111,465.45	183.47	3.80
BGWO—Vanilla Exploration	111,784	111,389.61	166.71	3.87
BGWO—Binary-Simple	111,960	111,484.71	210.11	3.78
BGWO—Binary-Complex	111,803	111,412.16	209.34	3.85
BDE—Vanilla Exploitation	113,731	112,748.32	449.92	2.69
BDE—Vanilla Exploration	113,456	112,854.19	298.97	2.60
BDE—Binary-Simple	114,342	113,489.10	381.88	2.05
BDE—Binary-Complex	114,152	113,496.81	334.71	2.05

Table 8. Mean relative optimality gap (%) across all 27 instances, with each instance weighted equally. These are descriptive averages of per-instance mean gaps, not gaps calculated from pooled raw fitness.

Algorithm	V-Exploit	V-Explore	Bin-Simple	Bin-Complex
BPSO	2.44	1.64	1.73	1.72
GA	0.41	1.75	0.86	0.52
BGWO	2.73	2.79	2.77	2.77
BDE	1.51	0.94	0.75	0.77

The index-0 results reveal that the relative performance of the four variants depends on both the underlying metaheuristic and the benchmark instance. Consequently, no single fixed configuration consistently dominates across all experimental conditions.

For BPSO, the exploration-oriented configuration generally provides stronger performance than the fixed exploitation profile as problem size increases. On *mknapcb5* [0], Vanilla-Exploration achieves a mean gap of 1.52%, compared with 2.55% for Vanilla-Exploitation, while Binary-Simple and Binary-Complex remain close to the exploration-oriented baseline, with gaps of 1.63% and 1.69%, respectively. A similar pattern is observed on *mknapcb9* [0], where Vanilla-Exploration, Binary-Simple, and Binary-Complex obtain gaps of 3.17%, 3.18%, and 3.20%, respectively, compared with 3.98% for Vanilla-Exploitation. Thus, the adaptive variants avoid the comparatively weaker behavior of the fixed exploitation profile on these larger BPSO instances while remaining competitive with the stronger fixed configuration.

GA exhibits a different pattern. Vanilla-Exploitation performs particularly well on the larger representative instances, achieving gaps of 0.60% on *mknapcb5* [0] and 0.89% on *mknapcb9* [0]. Binary-Complex obtains the strongest performance among the remaining variants on both instances, with gaps of 0.84% and 1.38%, respectively, but does not surpass the favorable fixed exploitation configuration. At the same time, Binary-Complex substantially improves upon Vanilla-Exploration on *mknapcb9* [0], reducing the mean gap from 3.24% to 1.38%. These results illustrate that the effectiveness of adaptive control depends on the behavior of the underlying metaheuristic and that a favorable fixed configuration can remain difficult to outperform in specific conditions.

For BGWO, differences among the four variants are comparatively small. On *mknapcb5* [0], mean gaps range from 2.73% to 2.91%, whereas on *mknapcb9* [0] they range from 3.78% to 3.87%. Binary-Simple achieves the lowest mean gap on *mknapcb9* [0], while Vanilla-Exploitation performs best on *mknapcb5* [0]. The comparatively narrow performance range suggests that configuration changes have a more limited effect on BGWO under the evaluated conditions.

The clearest benefits of trajectory-driven adaptation among the representative instances are observed for BDE. On *mknapcb5* [0], Binary-Complex achieves the lowest mean gap (0.74%), improving upon Vanilla-Exploitation (1.34%), Vanilla-Exploration (0.94%), and Binary-Simple (0.87%). On *mknapcb9* [0], both adaptive strategies achieve a gap of approximately 2.05%, compared with 2.69% and 2.60% for Vanilla-Exploitation and Vanilla-Exploration, respectively. This behavior indicates that BDE benefits particularly from dynamically alternating between the two predefined configurations under the proposed trajectory-driven control mechanism.

Overall, the descriptive results support two main observations. First, the preferred fixed configuration varies across metaheuristics and benchmark conditions, reinforcing the motivation for online configuration control rather than reliance on a single predefined search profile. Second, the trajectory-driven variants remain competitive across the evaluated conditions and provide clear improvements in several cases, particularly for BDE and for GA when compared with its exploration-oriented baseline. The following statistical analysis examines whether these observed differences are supported by inferential evidence.

In the published MKP evaluation, the LBLP strategy reached 20 of 30 known optima, while QPSO was reported to attain 21 of 30 best-known values [32]. In our study, Binary-Complex achieved a mean gap of 0.74% for BDE on *mknapcb5* [0], compared with 0.94% for the best fixed profile; on *mknapcb9* [0], the corresponding gaps were 2.05% and 2.60%. These outcomes provide context only: the benchmark cases, algorithms, reported metrics, and evaluation budgets differ, so they do not establish which method performs better.

To illustrate search-trajectory progression for the two binary variants, Figures 2 and 3 show the three *mknapcb1* instances, with indices 0, 15, and 29 from top to bottom. For each plotted variant, metaheuristic, and instance, the curve is the run with the best final fitness among that condition's 31 independent runs. These selected runs illustrate trajectories only: they are neither means nor typical behavior and provide no evidence of causal effects or method superiority; statistical conclusions continue to rely on all 31 runs per condition. Corresponding selected-run plots for *mknapcb5* and *mknapcb9* are provided in Appendix B.

For conciseness and readability, convergence illustrations are limited to three benchmark groups (*mknapcb1*, *mknapcb5*, and *mknapcb9*), with three instances per group. Displaying trajectories for every experimental condition would substantially expand the manuscript and supplementary material. This restriction concerns only the graphical illustrations; the tabulated descriptive and statistical analyses cover all 27 evaluated instances. Additional convergence plots can be prepared upon request.

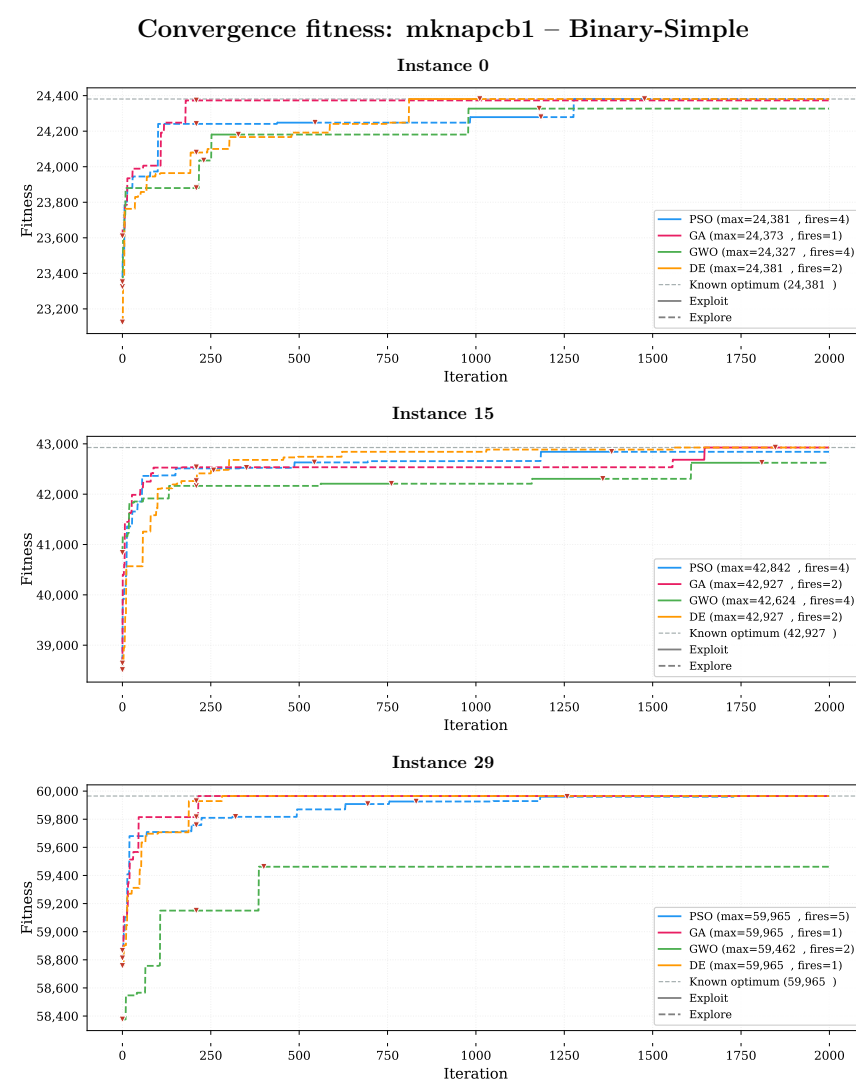


Figure 2. Selected Binary-Simple convergence trajectories for *mknapcb1*; instances 0, 15, and 29 appear from top to bottom. For each metaheuristic and instance, the curve is the best-final-fitness run among 31 independent runs.

6.2. Statistical Analysis

The expanded analysis uses all 27 selected instances and the four metaheuristics, yielding 108 metaheuristic–instance conditions. Within each condition, Binary-Complex is compared with the three alternative strategies using the two-sided Wilcoxon signed-rank

test on the 31 stored fitness values, paired by their common run order. Table 9 summarizes the direction of these comparisons by metaheuristic; the complete condition-level values appear in Appendix B. The individual Wilcoxon p -values are unadjusted. With 324 pairwise tests, these local counts are exploratory and must not be interpreted as family-wise-error-controlled evidence. A separate Bonferroni-Dunn post-hoc analysis is applied to the global mean ranks below.

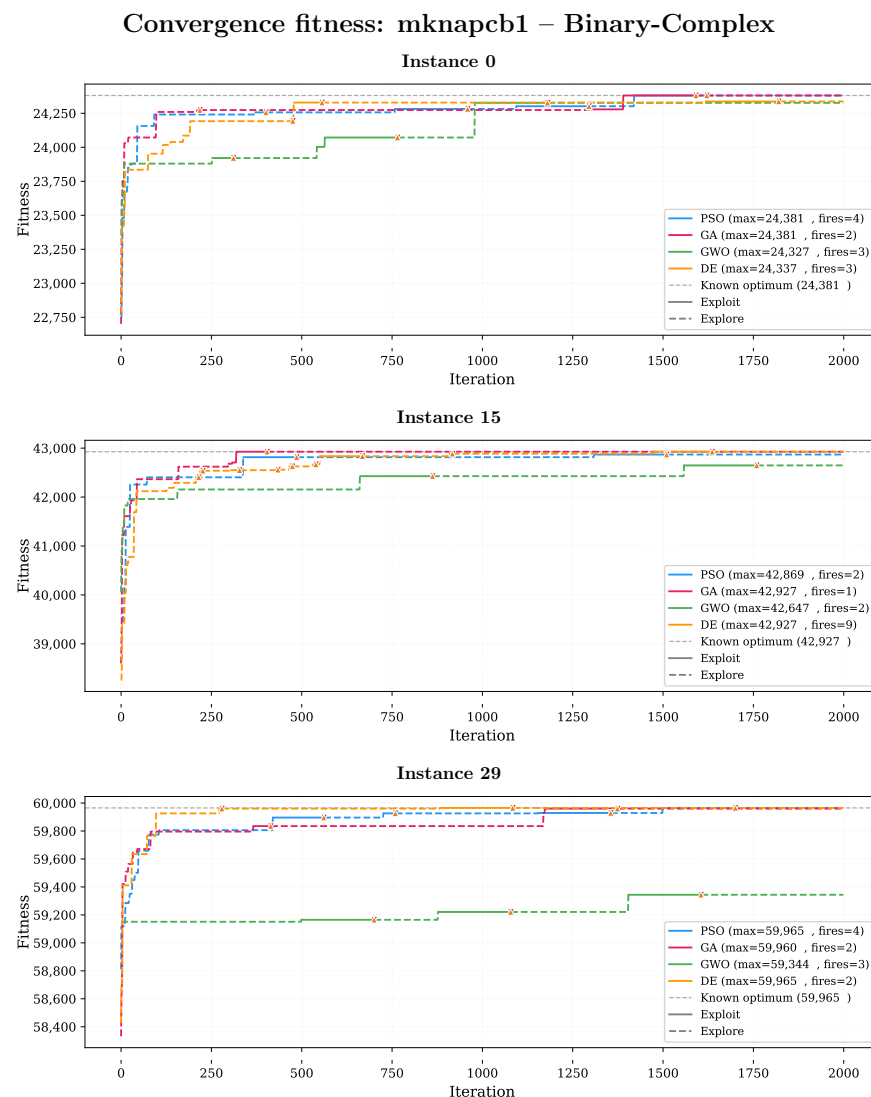


Figure 3. Selected Binary-Complex convergence trajectories for mknapcb1; instances 0, 15, and 29 appear from top to bottom. For each metaheuristic and instance, the curve is the best-final-fitness run among 31 independent runs.

Table 9. Exploratory, unadjusted Wilcoxon comparisons of Binary-Complex against each comparator ($\alpha = 0.05$, 31 run-index pairs per condition). Entries are favorable/not significant/unfavorable counts. Each algorithm row covers 27 instances; the total covers 108 conditions. No significant difference does not imply statistical equivalence.

Algorithm	vs. V-Exploit	vs. V-Explore	vs. Bin-Simple
BPSO	27/0/0	0/15/12	1/26/0
GA	3/4/20	22/5/0	17/10/0
BGWO	5/15/7	3/24/0	2/25/0
BDE	27/0/0	15/10/2	1/26/0
Total	62/19/27	40/54/14	21/87/0

Against Vanilla-Exploitation, Binary-Complex has 62 favorable, 19 non-significant, and 27 unfavorable local comparisons. Against Vanilla-Exploration, the corresponding counts are 40, 54, and 14. Against Binary-Simple, there are 21 favorable and 87 non-significant comparisons, with no unfavorable result at the unadjusted threshold. The benefits remain heterogeneous: BPSO and BDE have favorable comparisons against Vanilla-Exploitation on all 27 instances, whereas GA has 20 unfavorable comparisons against that fixed configuration. BPSO has no favorable comparison against Vanilla-Exploration and 12 unfavorable ones. The 21 favorable comparisons against Binary-Simple are concentrated in GA (17), with one each in BPSO and BDE and two in BGWO.

The Friedman analysis (Table 10) ranks the mean final fitness of the four strategies within each of the 108 metaheuristic–instance blocks; larger fitness receives the better rank and tied means receive average ranks. These blocks represent 27 distinct MKP instances evaluated with four algorithms, not 108 distinct optimization problems.

Table 10. Global Friedman ranking across the 108 metaheuristic–instance combinations.

Strategy	Mean Rank (1=Best)	Placement
Binary-Complex	2.13	1
Binary-Simple	2.39	2
Vanilla-Exploration	2.56	3
Vanilla-Exploitation	2.93	4

Friedman statistic: $\chi^2 = 21.64$, $p = 7.73 \times 10^{-5}$.

With the enlarged evaluation, the Friedman test detects an overall strategy difference ($\chi^2 = 21.64$, $p = 7.73 \times 10^{-5}$), replacing the non-significant result from the original nine-instance evaluation. Binary-Complex has the best mean rank (2.13), followed by Binary-Simple (2.39), Vanilla-Exploration (2.56), and Vanilla-Exploitation (2.93). An overall rejection does not show that every strategy differs from every other strategy; the relevant control-versus-baseline comparisons are reported in Table 11.

Table 11. Bonferroni-Dunn post-hoc comparisons with Binary-Complex as control across 108 blocks. Three two-sided comparisons are corrected using Bonferroni; $SE = 0.1757$ and $CD = 0.421$ at $\alpha = 0.05$. These adjusted p -values are separate from the unadjusted run-level Wilcoxon values.

Comparator	Rank Diff.	z	p_{unadj}	p_{Bonf}
Binary-Simple	0.26	1.48	0.1400	0.4200
Vanilla-Exploration	0.43	2.42	0.0153	0.0460
Vanilla-Exploitation	0.80	4.53	5.83×10^{-6}	1.75×10^{-5}

The adjusted comparisons favor Binary-Complex over Vanilla-Exploitation ($p_{\text{Bonf}} = 1.75 \times 10^{-5}$) and Vanilla-Exploration ($p_{\text{Bonf}} = 0.0460$), but do not establish a difference from Binary-Simple ($p_{\text{Bonf}} = 0.4200$). Thus, additional controller complexity has favorable local outcomes in selected conditions, but is not demonstrated to improve the global rank relative to the simpler adaptive controller at the adjusted threshold. These results are restricted to the sampled MKP cases and tested configurations; they do not establish universal superiority, performance on other problem domains, or the isolated contribution of DDTW.

6.3. Computational Overhead of Adaptive Control

Execution time is assessed from the 31 recorded runtime values for each strategy, metaheuristic, and instance. The complete mean and sample standard deviation in seconds for all 108 conditions appear in Appendix C. To summarize the enlarged sample, Table 12

reports the average relative runtime difference of Binary-Complex from each comparator. For each instance, the difference is $100(\bar{t}_{\text{Complex}}/\bar{t}_{\text{base}} - 1)$; the 27 resulting ratios are then averaged with equal instance weights. These descriptive averages are not confidence intervals or estimates from pooling unlike problem sizes.

Table 12. Mean per-instance runtime difference (%) of Binary-Complex across 27 instances. Positive values indicate longer runtime; negative values indicate shorter runtime.

Algorithm	vs. V-Exploit	vs. V-Explore	vs. Bin-Simple
BPSO	+11.57	+11.61	+0.68
GA	+24.69	+13.50	−0.09
BGWO	+11.39	+12.01	−0.04
BDE	+4.89	+4.67	−5.64

Relative to the fixed configurations, Binary-Complex takes approximately 11–12% longer on average for BPSO and BGWO. GA’s mean difference is 24.69% versus Vanilla-Exploitation and 13.50% versus Vanilla-Exploration. For BDE, the corresponding averages are 4.89% and 4.67%. Thus, the enlarged sample does not support a blanket claim that adaptive control has negligible cost relative to fixed profiles.

The incremental cost of Binary-Complex relative to Binary-Simple is much smaller: the mean relative difference is +0.68% for BPSO, −0.09% for GA, −0.04% for BGWO, and −5.64% for BDE. This last empirical runtime reduction is not evidence of a lower asymptotic complexity or of a changed iteration budget. The results show that trajectory analysis has an algorithm-dependent runtime cost, while the more elaborate controller generally adds little runtime relative to the simpler trajectory-driven strategy.

6.4. Adaptive Controller Behavior

Figure 4 summarizes the recorded transitions to the exploration-oriented configuration ($s_t = 1$) for the expanded campaign. For each benchmark group and metaheuristic, the three selected instances contribute $3 \times 31 = 93$ run-level transition counts per strategy. The bars are pooled means over those 93 counts; error bars are pooled sample standard deviations and therefore include both run-to-run and between-instance variation. They are not confidence intervals.

Across all 27 instances, the mean transition counts per run for Binary-Simple and Binary-Complex are, respectively, 3.65 and 4.80 for BPSO, 1.43 and 1.21 for GA, 2.12 and 1.51 for BGWO, and 3.71 and 7.19 for BDE. Each of these descriptive means is based on $27 \times 31 = 837$ run-level counts. The enlarged sample therefore shows more exploratory transitions for Binary-Complex in BPSO and BDE, but fewer in GA and BGWO.

The trajectory-analysis settings are unchanged across metaheuristics and instances. Nevertheless, the controllers do not produce a uniform switching schedule. Binary-Simple reacts to the plateau-distance condition, whereas Binary-Complex additionally uses multiple search conditions, persistence, and asymmetric switching.

These activity summaries should be considered alongside, rather than substituted for, the solution-quality results. For example, BDE’s greater activity with Binary-Complex coexists with a slightly lower average gap for Binary-Simple (0.75% versus 0.77%). More transitions are therefore not inherently beneficial. The aggregate counts describe an association between the controller, algorithm, and observed trajectory; they do not identify a causal mechanism or replace synchronized temporal diagnostics.

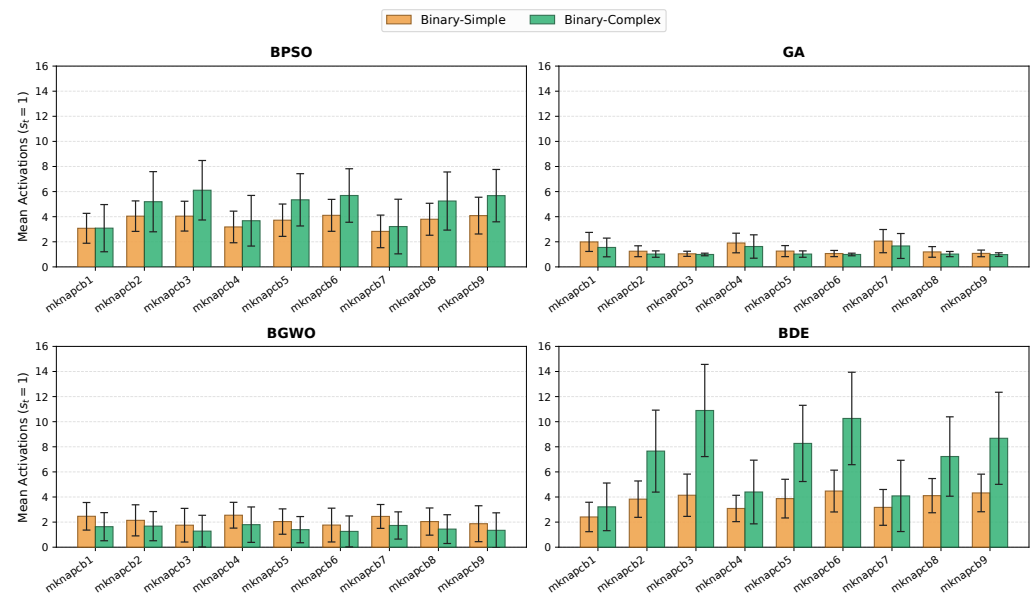


Figure 4. Exploration-oriented transitions for Binary-Simple and Binary-Complex across the nine MKP groups, using all 27 selected instances. Each bar pools indices 0, 15, and 29 (93 runs per group, metaheuristic, and strategy); error bars are pooled sample standard deviations.

Figure 5 supplies one such synchronized trace. It is a single Binary-Complex run of BDE on the first instance of mknapcb1 ($n = 100$, $m = 5$), with seed $q = 1$ and the parameterization in Table 4. The run reaches a best-so-far fitness of 24,273 (gap 0.44% from the known optimum 24,381) and enters the exploration-oriented configuration three times. Panel (a) shows f_t^* with the controller state s_t in the background. The blue frame marks the rising window, and the orange frame the flat window, both drawn in panel (b) after shifting each window to start at zero. There, the observed window is compared with the sustained-progress reference R_t (slope $s_{\min} = 2$) and the constant reference C_t . The rising window, ending at $t = 308$, gains 368 fitness units, against 398 for the ramp template, and does so as a staircase rather than a straight line. The flat window, ending at $t = 1623$, lies on C_t while R_t diverges. DDTW is applied to the first differences of these series, so R_t is a fixed-slope template, not a fitted line. Panel (c) shows $D_R(t)$ and $D_C(t)$ with the history-based thresholds $\theta_R(t)$ and $\theta_C(t)$. Panel (d) shows $\Delta_t = D_R(t) - D_C(t)$ with its threshold $\theta_\Delta(t)$, and the state s_t . For BDE, $s_t = 0$ sets $(F, CR) = (0.5, 0.9)$ and $s_t = 1$ sets $(F, CR) = (0.9, 0.3)$. Triangles mark transitions between the two configurations. Distances are defined once the window is full ($t = 200$); percentile thresholds begin at $t = 209$. These are the quantities Binary-Complex uses at each iteration. The trace illustrates the controller for one algorithm and one instance. It does not establish that the same references characterize every metaheuristic, and it does not replace the aggregate counts in Figure 4.

6.5. OAT Sensitivity Analysis

To examine the influence of the principal trajectory-monitor settings, we conducted a one-factor-at-a-time (OAT) sensitivity analysis. For each configuration, one parameter was changed while the remaining parameters were held at their paper values. The analysis covered the observation window W , the Sakoe–Chiba band, the lower and upper percentiles used for the adaptive thresholds, the plateau-improvement limit $\pi_{\max}$, and the persistence parameter τ_{pat} . The reference slope $\text{min_slope} = 2.0$ was held fixed. Each parameter was evaluated at its paper value and three alternatives, yielding 19 configurations per strategy.

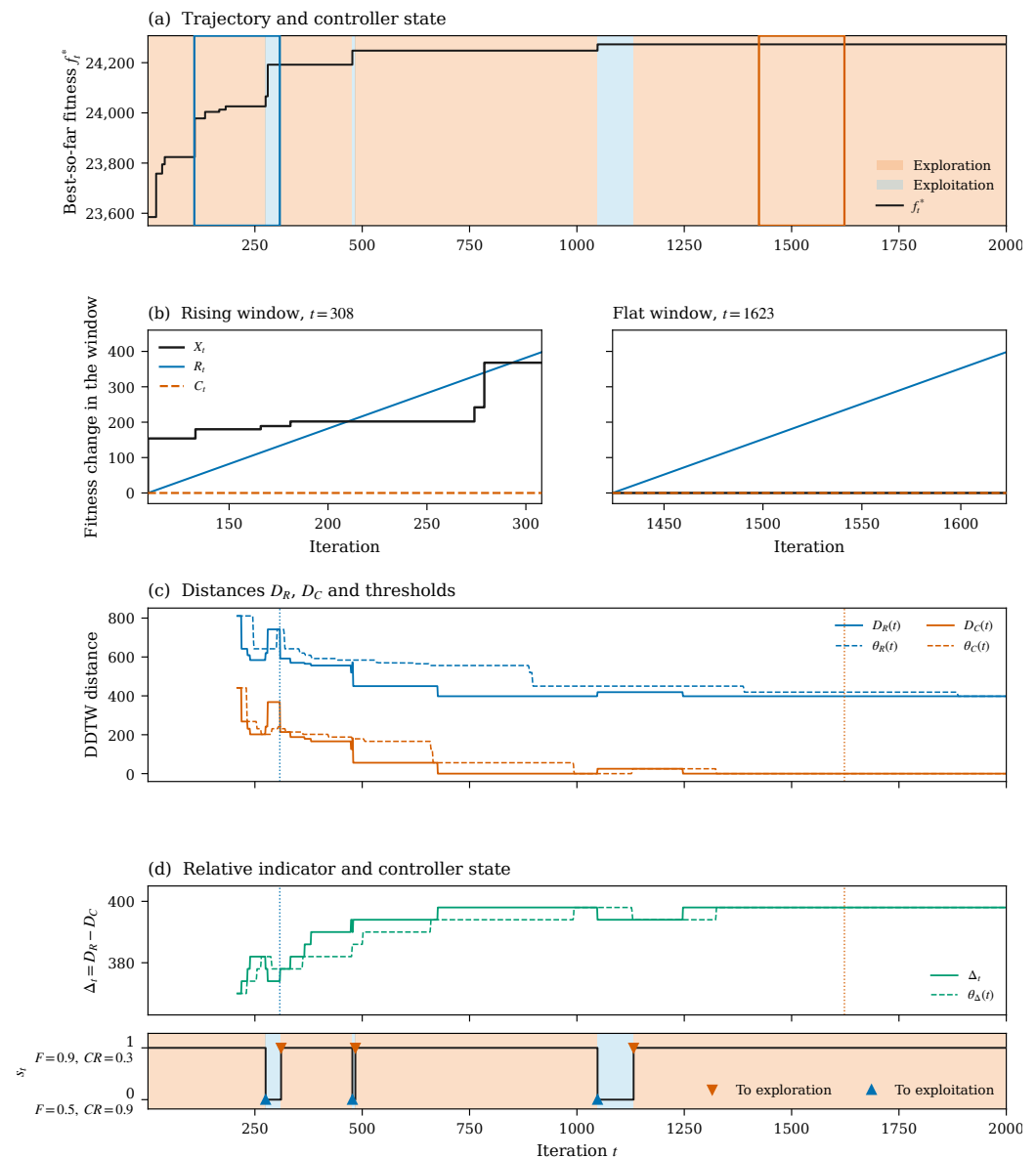


Figure 5. Temporal diagnostics for one Binary-Complex run (BDE, mknapcb1, seed $q = 1$).

Two separate campaigns evaluated Binary-Complex and Binary-Simple. Each campaign used the three selected instances mknapcb1 [0, 15, 29], the four metaheuristics (PSO, GA, GWO, and DE), and 31 independent runs per configuration, instance, and metaheuristic. The audit found all 19 expected configurations and all 228 expected result files in each campaign; all 7068 serialized fitness values per campaign were finite. Table 13 summarizes the observed changes relative to the paper configuration. For each parameter, the reported range is the minimum and maximum signed Δ across its three alternatives; Δ is the difference in mean fitness relative to the base, aggregated across the four metaheuristics, as reported by the OAT analysis. Positive values indicate higher fitness and negative values lower fitness. These ranges are descriptive summaries, not confidence intervals or measures of run-to-run variability.

The observed effects were not uniform across parameter values. For Binary-Complex, the signed differences changed direction for several parameters, so these data do not identify one universally best setting. For Binary-Simple, the band, upper-percentile, plateau-limit, and persistence alternatives yielded the same aggregate value as the base in the reported table. Upper-percentile, plateau-limit, and persistence settings do not enter the

Binary-Simple decision rule. Its invariance to the band also follows from Equation (17), rather than serving as independent evidence of robustness. These observations do not establish insensitivity of Binary-Complex, whose ramp-based signal can depend on the band.

Table 13. Descriptive OAT sensitivity results. Each entry gives the range of signed mean-fitness differences (Δ) from the paper configuration across the three alternative values of that parameter.

Parameter (Base)	Alternative Values	Binary-Complex Δ	Binary-Simple Δ
Window W (200)	50, 100, 400	−2.780 to −0.484	−2.132 to +4.801
Sakoe–Chiba band (2)	1, 4, 8	−0.737 to +0.113	0.000
Lower percentile (40)	20, 30, 50	−0.008 to +2.777	−0.599 to +4.180
Upper percentile (60)	50, 70, 80	−1.183 to +1.419	0.000
Plateau limit $\pi_{\max}$ (5)	3, 8, 12	−0.129 to +0.304	0.000
Persistence τ_{pat} (3)	1, 2, 5	−3.970 to +2.038	0.000

The evidence is limited to three indices from one benchmark group and does not establish sensitivity across the remaining MKP groups, parameter interactions, or the fixed reference slope. The aggregate summaries also do not provide paired base-versus-alternative differences by run, algorithm, and index, uncertainty intervals, or multiplicity-adjusted tests. Therefore, we report this OAT study as a bounded descriptive exploration, not as statistical evidence of robustness or parameter optimality. In particular, the table’s minimum–maximum ranges cannot be converted into standard deviations, standard errors, or confidence intervals. Such estimates require the original run-level data, not the six reported aggregate ranges.

An inferential extension would retain algorithm and instance strata and estimate, for each alternative, the mean paired fitness difference from its base together with a run-level uncertainty interval, provided that matching seeds are verified. If seed correspondence cannot be established, an independent-sample uncertainty analysis is required instead. Multiple parameter contrasts would need an explicit multiplicity correction. This extension, variation of $s_{\min}$, and replication on other MKP groups have not been performed in the evidence reported here. Therefore neither reference-pattern adequacy across algorithms nor stability under profile or slope changes is claimed.

7. Conclusions and Future Work

This study evaluated trajectory-driven configuration control for BPSO, GA, BGWO, and BDE on 27 MKP instances: three indices from each of nine benchmark groups, with $n \in \{100, 250, 500\}$ items and $m \in \{5, 10, 30\}$ constraints. The DDTW-based framework compares recent best-so-far fitness trajectories with progress and plateau references to switch between predefined exploration and exploitation configurations. Binary-Simple and Binary-Complex differ in the complexity of this control.

The experimental results show that the effectiveness of a fixed parameter configuration depends on both the underlying metaheuristic and the problem instance. Neither the exploration nor the exploitation-oriented configuration consistently provided the best performance across all experimental conditions. This variability supports the motivation for online configuration control based on information generated during the search rather than relying exclusively on a single predefined operating configuration.

The enlarged evaluation contains 13,392 runs across 432 experimental conditions. Binary-Complex achieves the best global mean rank (2.13), followed by Binary-Simple (2.39), Vanilla-Exploration (2.56), and Vanilla-Exploitation (2.93). Unlike the original nine-instance analysis, the Friedman test detects an overall strategy difference ($p = 7.73 \times 10^{-5}$). Bonferroni–Dunn post-hoc comparisons favor Binary-Complex over the two fixed configurations, but do not establish a difference from Binary-Simple ($p_{\text{Bonf}} = 0.4200$). The unadjusted

run-level Wilcoxon analysis identifies 21 favorable comparisons against Binary-Simple and no unfavorable ones across 108 conditions; these exploratory counts do not override the adjusted global comparison or establish a universal benefit from added complexity.

The effect of trajectory-driven adaptation was nevertheless metaheuristic-dependent. Particularly favorable results were observed for BDE, where the adaptive strategies improved solution quality over the fixed configurations in several benchmark instances. BPSO also showed that the adaptive variants can remain competitive with a favorable fixed exploration-oriented configuration while avoiding exclusive reliance on that search profile. In contrast, GA exhibited conditions in which the fixed exploitation-oriented configuration remained particularly strong, while BGWO showed comparatively smaller differences among configurations. These findings indicate that trajectory-driven control should not be interpreted as a mechanism that guarantees improvement for every metaheuristic and problem instance, but rather as a common feedback principle whose effect depends on the search dynamics generated by the underlying algorithm.

The expanded controller-activity analysis shows more exploratory transitions with Binary-Complex for BPSO and BDE on average, and fewer for GA and BGWO. These aggregate patterns do not demonstrate that more switches cause better performance. Runtime comparisons across all 27 instances show measurable overhead relative to fixed profiles; however, the added controller complexity contributes little runtime relative to Binary-Simple for BPSO, GA, and BGWO, and shorter empirical runtime for BDE.

The results support explicit pattern-based analysis of the search trajectory as a viable source of feedback for online configuration control. Rather than establishing a universally superior parameter configuration, the proposed framework provides a common mechanism for adapting the operating behavior of different metaheuristics according to information extracted from their evolving convergence trajectories.

7.1. Limitations

Despite the encouraging results, the present study has several limitations. First, the adaptive controller is restricted to discrete switching between two predefined configurations representing exploration- and exploitation-oriented search behaviors. Although this formulation provides a clear and controlled setting for evaluating trajectory-driven adaptation, it restricts the controller to a small set of possible operating states and does not consider intermediate parameter configurations. Second, the trajectory-analysis mechanism relies exclusively on the recent best-so-far fitness trajectory. This provides an algorithm-independent temporal signal, but it does not explicitly capture other aspects of the search state, such as population diversity, spatial distribution, or changes in the structure of the candidate solutions. Consequently, distinct population states may potentially generate similar convergence trajectories. Third, the experimental study is restricted to the MKP and to four population-based metaheuristics. Although the same trajectory-analysis mechanism and parameterization were evaluated across different metaheuristic paradigms and benchmark dimensions, broader conclusions regarding cross-problem applicability, require validation on additional combinatorial optimization problems and algorithmic families.

The reference patterns and trajectory-analysis parameters were kept fixed throughout the main 27-instance campaign. This design facilitates a controlled assessment of the proposed mechanism, but the sensitivity of the controller to the trajectory window, reference-pattern characteristics, percentile thresholds, and persistence parameters remain to be studied systematically. The separate OAT analysis in Section 6.5 is limited to three indices from `mknpcb1`; it does not establish sensitivity across all nine groups or quantify parameter interactions.

7.2. Future Research Directions

The present results suggest several directions for extending trajectory-driven adaptive configuration control. A first direction is the development of continuous configuration-control mechanisms in which the trajectory information is mapped to intermediate parameter configurations rather than exclusively triggering discrete transitions between two predefined states. Such mechanisms could provide finer control over the exploration–exploitation balance while preserving the trajectory-driven feedback principle introduced in this work. On the other hand, a second direction is to enrich the characterization of the search state by combining trajectory-pattern information with complementary population-level signals. Measures of population diversity, spatial distribution, improvement rate, or other search descriptors could be integrated with the temporal information extracted from the convergence trajectory. This would enable the controller to distinguish search conditions that may exhibit similar fitness trajectories but different population dynamics.

Future work should also evaluate the framework on additional discrete and continuous optimization problems and with other metaheuristic families. Such experiments would allow the cross-problem applicability of trajectory-driven control to be assessed and would help determine which properties of the underlying search dynamics favor this form of adaptation. The trajectory-analysis principle considered here provides a basis for investigating more general forms of search-state estimation and algorithm-level decision-making. Future extensions may use trajectory-based search-state information not only to adapt the internal configuration of a single metaheuristic, but also to coordinate multiple search processes, select among alternative metaheuristics, or determine when information should be exchanged between complementary optimization strategies. These directions extend the role of trajectory analysis from parameter configuration control toward broader adaptive and cooperative metaheuristic frameworks.

Author Contributions: Conceptualization and proposed approach, J.G., J.V., and E.V.; methodology, J.V., L.E., and E.V.; software, J.V. and L.E.; formal analysis, E.V., J.V., and J.G.; investigation, E.V., J.V., and J.G.; validation, R.S. and J.G.; writing—original draft preparation, E.V. and J.V.; writing—review and editing, E.V., L.E. and J.V. All authors have read and agreed to the published version of the manuscript.

Funding: Emanuel Vega was supported by the Agencia Nacional de Investigación y Desarrollo (ANID) under FONDECYT Initiation Project 11261348.

Informed Consent Statement: Not applicable.

Data Availability Statement: The MKP benchmark files analyzed in this study are the Chu–Beasley OR-Library instances. The development repository is https://github.com/jochevipe/DTW_optimization (accessed on 25 September 2026). No permanently archived public release of the full experimental code and result data is identified here. The campaign’s aggregate files do not contain full controller traces, and the OAT table does not provide the run-level data needed for uncertainty estimation.

Acknowledgments: The authors extend their appreciation to the Intelligent Industry Doctorate Program and the Doctoral Program from the School of Informatics, both from the Pontificia Universidad Católica de Valparaíso for supporting this work. Additionally, the authors extend their gratitude to the PUCV Library for providing access to essential documents and resources.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Supplementary Solution Quality Results

This appendix reports the 24 selected instances not shown in the three representative index-0 tables in Section 6.1. Together, the main-text and supplementary tables cover all

27 instances from indices 0, 15, and 29 of the nine MKP groups. Each table reports the best fitness, mean final fitness, sample standard deviation, and relative optimality gap over 31 independent runs for the four metaheuristics and four strategies. All table values were checked against the corresponding serialized fitness arrays.

Appendix A.1. Benchmark Group *mknapcb1*

Table A1. Solution quality on *mknapcb1* (inst. 15, $n = 100$, $m = 5$, $f^* = 42,927$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	42,657	42,376.13	112.14	1.28
BPSO—Vanilla Exploration	42,927	42,730.74	69.38	0.46
BPSO—Binary-Simple	42,842	42,704.97	68.28	0.52
BPSO—Binary-Complex	42,869	42,725.26	66.05	0.47
GA—Vanilla Exploitation	42,927	42,804.35	73.06	0.29
GA—Vanilla Exploration	42,869	42,702.42	83.65	0.52
GA—Binary-Simple	42,927	42,753.61	87.57	0.40
GA—Binary-Complex	42,927	42,796.29	82.89	0.30
BGWO—Vanilla Exploitation	42,624	42,395.35	86.08	1.24
BGWO—Vanilla Exploration	42,647	42,348.19	122.62	1.35
BGWO—Binary-Simple	42,624	42,356.42	91.12	1.33
BGWO—Binary-Complex	42,647	42,390.97	119.39	1.25
BDE—Vanilla Exploitation	42,886	42,676.90	134.63	0.58
BDE—Vanilla Exploration	42,927	42,848.68	73.08	0.18
BDE—Binary-Simple	42,927	42,853.19	58.41	0.17
BDE—Binary-Complex	42,927	42,843.94	82.17	0.19

Table A2. Solution quality on *mknapcb1* (inst. 29, $n = 100$, $m = 5$, $f^* = 59,965$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	59,960	59,917.65	33.92	0.08
BPSO—Vanilla Exploration	59,960	59,955.03	11.81	0.02
BPSO—Binary-Simple	59,965	59,950.81	16.00	0.02
BPSO—Binary-Complex	59,965	59,953.35	13.06	0.02
GA—Vanilla Exploitation	59,960	59,913.29	35.28	0.09
GA—Vanilla Exploration	59,960	59,894.48	47.95	0.12
GA—Binary-Simple	59,965	59,931.03	34.69	0.06
GA—Binary-Complex	59,960	59,929.13	41.17	0.06
BGWO—Vanilla Exploitation	59,416	59,167.39	100.47	1.33
BGWO—Vanilla Exploration	59,340	59,186.42	90.56	1.30
BGWO—Binary-Simple	59,462	59,192.84	104.92	1.29
BGWO—Binary-Complex	59,344	59,188.16	98.32	1.30
BDE—Vanilla Exploitation	59,960	59,918.13	51.39	0.08
BDE—Vanilla Exploration	59,965	59,960.32	1.25	0.01
BDE—Binary-Simple	59,965	59,959.32	4.78	0.01
BDE—Binary-Complex	59,965	59,960.16	0.90	0.01

Appendix A.2. Benchmark Group *mknapcb2*

Table A3. Solution quality on *mknapcb2* (inst. 0, $n = 250$, $m = 5$, $f^* = 59,312$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	58,694	58,320.39	147.28	1.67
BPSO—Vanilla Exploration	59,000	58,817.29	95.05	0.83
BPSO—Binary-Simple	58,947	58,762.52	79.06	0.93
BPSO—Binary-Complex	59,131	58,765.71	121.62	0.92

Table A3. *Cont.*

MH—Variant	Best Fitness	Average	Std.	Gap (%)
GA—Vanilla Exploitation	59,226	59,136.48	61.81	0.30
GA—Vanilla Exploration	58,957	58,741.10	104.44	0.96
GA—Binary-Simple	59,186	58,919.13	166.92	0.66
GA—Binary-Complex	59,194	59,030.42	100.03	0.47
BGWO—Vanilla Exploitation	58,504	58,309.87	115.38	1.69
BGWO—Vanilla Exploration	58,686	58,249.55	154.95	1.79
BGWO—Binary-Simple	58,531	58,269.39	136.12	1.76
BGWO—Binary-Complex	58,686	58,241.84	137.10	1.80
BDE—Vanilla Exploitation	59,060	58,768.68	185.62	0.92
BDE—Vanilla Exploration	59,162	59,024.42	68.67	0.48
BDE—Binary-Simple	59,235	59,093.00	66.73	0.37
BDE—Binary-Complex	59,266	59,102.74	81.43	0.35

Table A4. Solution quality on mknpcb2 (inst. 15, $n = 250$, $m = 5$, $f^* = 110,256$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	106,374	105,320.90	441.49	4.48
BPSO—Vanilla Exploration	107,809	107,279.29	254.25	2.70
BPSO—Binary-Simple	107,584	107,132.13	300.58	2.83
BPSO—Binary-Complex	107,863	107,158.10	315.71	2.81
GA—Vanilla Exploitation	110,167	110,080.61	49.07	0.16
GA—Vanilla Exploration	108,057	107,421.48	336.50	2.57
GA—Binary-Simple	110,049	109,117.06	481.46	1.03
GA—Binary-Complex	110,095	109,849.87	447.72	0.37
BGWO—Vanilla Exploitation	107,193	106,435.23	318.64	3.47
BGWO—Vanilla Exploration	107,108	106,207.55	434.30	3.67
BGWO—Binary-Simple	108,034	106,278.61	482.64	3.61
BGWO—Binary-Complex	107,616	106,263.39	410.61	3.62
BDE—Vanilla Exploitation	108,908	107,911.77	697.32	2.13
BDE—Vanilla Exploration	109,776	109,367.52	296.86	0.81
BDE—Binary-Simple	109,850	109,527.23	197.69	0.66
BDE—Binary-Complex	109,838	109,410.74	288.50	0.77

Table A5. Solution quality on mknpcb2 (inst. 29, $n = 250$, $m = 5$, $f^* = 154,662$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	154,187	153,982.32	105.02	0.44
BPSO—Vanilla Exploration	154,475	154,318.29	71.05	0.22
BPSO—Binary-Simple	154,480	154,261.03	82.54	0.26
BPSO—Binary-Complex	154,445	154,251.45	79.04	0.27
GA—Vanilla Exploitation	154,654	154,587.35	47.91	0.05
GA—Vanilla Exploration	154,307	154,074.48	116.31	0.38
GA—Binary-Simple	154,623	154,406.74	166.30	0.17
GA—Binary-Complex	154,623	154,509.23	103.04	0.10
BGWO—Vanilla Exploitation	151,991	151,397.19	312.63	2.11
BGWO—Vanilla Exploration	152,084	151,456.13	287.82	2.07
BGWO—Binary-Simple	152,084	151,484.71	296.56	2.05
BGWO—Binary-Complex	152,084	151,476.55	276.14	2.06
BDE—Vanilla Exploitation	154,595	154,364.55	147.32	0.19
BDE—Vanilla Exploration	154,654	154,601.87	28.26	0.04
BDE—Binary-Simple	154,652	154,610.03	34.37	0.03
BDE—Binary-Complex	154,654	154,606.52	31.16	0.04

Appendix A.3. Benchmark Group *mnknapcb3***Table A6.** Solution quality on *mnknapcb3* (inst. 0, $n = 500$, $m = 5$, $f^* = 120,130$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	117,152	116,590.00	250.28	2.95
BPSO—Vanilla Exploration	118,393	118,052.35	210.47	1.73
BPSO—Binary-Simple	118,421	117,935.90	207.34	1.83
BPSO—Binary-Complex	118,181	117,890.58	194.54	1.86
GA—Vanilla Exploitation	120,113	119,966.16	58.02	0.14
GA—Vanilla Exploration	118,541	117,907.84	301.96	1.85
GA—Binary-Simple	119,911	119,230.77	388.95	0.75
GA—Binary-Complex	120,000	119,870.13	77.47	0.22
BGWO—Vanilla Exploitation	118,046	117,610.81	194.98	2.10
BGWO—Vanilla Exploration	117,961	117,475.84	215.64	2.21
BGWO—Binary-Simple	117,939	117,502.03	204.21	2.19
BGWO—Binary-Complex	118,062	117,525.94	265.49	2.17
BDE—Vanilla Exploitation	118,648	118,145.74	369.30	1.65
BDE—Vanilla Exploration	119,095	118,742.87	244.04	1.15
BDE—Binary-Simple	119,714	119,199.97	298.48	0.77
BDE—Binary-Complex	119,692	119,212.68	293.76	0.76

Table A7. Solution quality on *mnknapcb3* (inst. 15, $n = 500$, $m = 5$, $f^* = 220,514$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	206,972	205,565.68	789.75	6.78
BPSO—Vanilla Exploration	210,600	209,477.42	522.59	5.00
BPSO—Binary-Simple	209,660	208,966.10	458.41	5.24
BPSO—Binary-Complex	210,213	208,905.10	585.05	5.26
GA—Vanilla Exploitation	220,405	220,334.39	45.63	0.08
GA—Vanilla Exploration	211,512	209,649.35	716.96	4.93
GA—Binary-Simple	220,283	218,225.87	2323.57	1.04
GA—Binary-Complex	220,339	220,170.00	125.05	0.16
BGWO—Vanilla Exploitation	211,573	210,454.29	462.48	4.56
BGWO—Vanilla Exploration	211,231	210,074.71	490.66	4.73
BGWO—Binary-Simple	211,231	210,330.29	485.51	4.62
BGWO—Binary-Complex	211,360	210,137.29	486.66	4.71
BDE—Vanilla Exploitation	214,395	211,759.23	1630.33	3.97
BDE—Vanilla Exploration	218,564	216,862.42	1090.83	1.66
BDE—Binary-Simple	218,854	216,966.32	1156.68	1.61
BDE—Binary-Complex	218,602	217,000.26	947.29	1.59

Table A8. Solution quality on *mnknapcb3* (inst. 29, $n = 500$, $m = 5$, $f^* = 299,904$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	297,777	297,378.16	202.32	0.84
BPSO—Vanilla Exploration	298,529	298,262.74	157.97	0.55
BPSO—Binary-Simple	298,444	298,151.42	144.18	0.58
BPSO—Binary-Complex	298,561	298,158.90	151.15	0.58
GA—Vanilla Exploitation	299,770	299,683.35	47.88	0.07
GA—Vanilla Exploration	298,118	297,751.48	213.48	0.72
GA—Binary-Simple	299,675	298,744.39	641.93	0.39
GA—Binary-Complex	299,689	299,550.97	86.20	0.12
BGWO—Vanilla Exploitation	294,838	292,111.71	792.91	2.60
BGWO—Vanilla Exploration	294,838	292,182.97	768.34	2.57
BGWO—Binary-Simple	294,838	292,183.87	756.72	2.57
BGWO—Binary-Complex	294,838	292,176.81	762.16	2.58

Table A8. Cont.

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BDE—Vanilla Exploitation	299,305	298,530.13	370.82	0.46
BDE—Vanilla Exploration	299,469	299,285.19	106.83	0.21
BDE—Binary-Simple	299,662	299,412.29	120.30	0.16
BDE—Binary-Complex	299,614	299,460.77	100.40	0.15

Appendix A.4. Benchmark Group *mnknapcb4*Table A9. Solution quality on mnknapcb4 (inst. 0, $n = 100$, $m = 10$, $f^* = 23,064$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	23,050	22,888.52	74.28	0.76
BPSO—Vanilla Exploration	23,055	22,983.94	53.72	0.35
BPSO—Binary-Simple	23,055	22,948.87	52.51	0.50
BPSO—Binary-Complex	23,055	22,968.35	58.21	0.41
GA—Vanilla Exploitation	23,057	22,876.87	90.41	0.81
GA—Vanilla Exploration	23,057	22,915.45	74.49	0.64
GA—Binary-Simple	23,057	22,911.58	70.00	0.66
GA—Binary-Complex	23,057	22,910.55	84.55	0.67
BGWO—Vanilla Exploitation	22,800	22,654.06	80.20	1.78
BGWO—Vanilla Exploration	22,780	22,645.32	60.10	1.82
BGWO—Binary-Simple	22,781	22,657.35	64.31	1.76
BGWO—Binary-Complex	22,850	22,651.42	81.21	1.79
BDE—Vanilla Exploitation	23,042	22,879.74	101.76	0.80
BDE—Vanilla Exploration	23,057	22,964.84	58.34	0.43
BDE—Binary-Simple	23,057	22,970.10	54.17	0.41
BDE—Binary-Complex	23,057	22,958.77	60.23	0.46

Table A10. Solution quality on mnknapcb4 (inst. 15, $n = 100$, $m = 10$, $f^* = 42,995$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	42,651	42,293.16	145.00	1.63
BPSO—Vanilla Exploration	42,814	42,669.13	91.23	0.76
BPSO—Binary-Simple	42,932	42,661.13	97.40	0.78
BPSO—Binary-Complex	42,788	42,654.65	80.86	0.79
GA—Vanilla Exploitation	42,995	42,770.13	86.65	0.52
GA—Vanilla Exploration	42,814	42,621.58	94.09	0.87
GA—Binary-Simple	42,995	42,723.35	90.95	0.63
GA—Binary-Complex	42,995	42,705.35	128.30	0.67
BGWO—Vanilla Exploitation	42,497	42,206.97	93.78	1.83
BGWO—Vanilla Exploration	42,417	42,168.52	111.69	1.92
BGWO—Binary-Simple	42,656	42,179.97	136.41	1.90
BGWO—Binary-Complex	42,614	42,195.39	144.96	1.86
BDE—Vanilla Exploitation	42,917	42,551.45	192.19	1.03
BDE—Vanilla Exploration	42,995	42,801.42	73.27	0.45
BDE—Binary-Simple	42,995	42,828.35	59.70	0.39
BDE—Binary-Complex	42,995	42,811.16	70.22	0.43

Table A11. Solution quality on mnknapcb4 (inst. 29, $n = 100$, $m = 10$, $f^* = 60,633$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	60,625	60,508.26	46.56	0.21
BPSO—Vanilla Exploration	60,633	60,609.65	25.27	0.04
BPSO—Binary-Simple	60,633	60,588.39	32.83	0.07
BPSO—Binary-Complex	60,633	60,589.29	36.90	0.07

Table A11. Cont.

MH—Variant	Best Fitness	Average	Std.	Gap (%)
GA—Vanilla Exploitation	60,633	60,525.48	60.07	0.18
GA—Vanilla Exploration	60,633	60,555.03	57.00	0.13
GA—Binary-Simple	60,633	60,547.03	52.55	0.14
GA—Binary-Complex	60,629	60,546.42	46.04	0.14
BGWO—Vanilla Exploitation	60,100	59,609.71	167.38	1.69
BGWO—Vanilla Exploration	60,284	59,680.00	187.05	1.57
BGWO—Binary-Simple	60,100	59,718.39	179.52	1.51
BGWO—Binary-Complex	59,932	59,696.32	112.34	1.54
BDE—Vanilla Exploitation	60,633	60,532.19	74.29	0.17
BDE—Vanilla Exploration	60,633	60,628.74	6.67	0.01
BDE—Binary-Simple	60,633	60,618.68	21.39	0.02
BDE—Binary-Complex	60,633	60,615.35	25.20	0.03

Appendix A.5. Benchmark Group *mknapcb5*Table A12. Solution quality on mknapcb5 (inst. 15, $n = 250$, $m = 10$, $f^* = 110,841$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	106,755	105,988.74	406.15	4.38
BPSO—Vanilla Exploration	108,332	107,597.61	329.14	2.93
BPSO—Binary-Simple	108,424	107,519.48	334.14	3.00
BPSO—Binary-Complex	108,405	107,572.19	296.19	2.95
GA—Vanilla Exploitation	110,638	110,345.13	163.74	0.45
GA—Vanilla Exploration	108,281	107,519.61	358.12	3.00
GA—Binary-Simple	110,514	109,117.52	624.95	1.55
GA—Binary-Complex	110,459	110,060.10	215.39	0.70
BGWO—Vanilla Exploitation	107,401	106,901.42	258.30	3.55
BGWO—Vanilla Exploration	107,268	106,729.35	267.47	3.71
BGWO—Binary-Simple	107,258	106,627.13	236.02	3.80
BGWO—Binary-Complex	107,934	106,833.48	355.12	3.62
BDE—Vanilla Exploitation	109,075	108,051.29	507.30	2.52
BDE—Vanilla Exploration	109,399	109,038.81	237.18	1.63
BDE—Binary-Simple	110,039	109,486.39	363.18	1.22
BDE—Binary-Complex	109,980	109,527.03	253.23	1.19

Table A13. Solution quality on mknapcb5 (inst. 29, $n = 250$, $m = 10$, $f^* = 149,704$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	148,988	148,630.97	129.86	0.72
BPSO—Vanilla Exploration	149,173	148,986.35	87.67	0.48
BPSO—Binary-Simple	149,102	148,953.97	83.92	0.50
BPSO—Binary-Complex	149,246	148,973.71	99.74	0.49
GA—Vanilla Exploitation	149,630	149,408.29	105.24	0.20
GA—Vanilla Exploration	149,235	148,746.32	144.58	0.64
GA—Binary-Simple	149,472	149,137.71	190.11	0.38
GA—Binary-Complex	149,548	149,200.00	198.33	0.34
BGWO—Vanilla Exploitation	146,693	145,872.97	304.29	2.56
BGWO—Vanilla Exploration	146,693	146,107.97	255.16	2.40
BGWO—Binary-Simple	146,693	146,062.00	275.82	2.43
BGWO—Binary-Complex	146,693	146,094.94	285.63	2.41
BDE—Vanilla Exploitation	149,484	149,116.61	199.60	0.39
BDE—Vanilla Exploration	149,431	149,278.10	79.35	0.28
BDE—Binary-Simple	149,570	149,389.32	107.31	0.21
BDE—Binary-Complex	149,549	149,399.29	83.80	0.20

Appendix A.6. Benchmark Group *mnknapcb6***Table A14.** Solution quality on *mnknapcb6* (inst. 0, $n = 500$, $m = 10$, $f^* = 117,726$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	114,418	113,745.42	350.35	3.38
BPSO—Vanilla Exploration	115,347	114,848.55	201.56	2.44
BPSO—Binary-Simple	115,128	114,685.23	227.20	2.58
BPSO—Binary-Complex	115,218	114,689.16	204.68	2.58
GA—Vanilla Exploitation	117,492	117,299.23	112.59	0.36
GA—Vanilla Exploration	115,553	114,756.71	323.00	2.52
GA—Binary-Simple	117,153	116,063.45	491.12	1.41
GA—Binary-Complex	117,348	117,010.52	445.41	0.61
BGWO—Vanilla Exploitation	114,725	113,975.13	274.89	3.19
BGWO—Vanilla Exploration	114,446	113,825.61	229.23	3.31
BGWO—Binary-Simple	114,465	113,822.39	281.92	3.32
BGWO—Binary-Complex	114,655	113,735.71	288.35	3.39
BDE—Vanilla Exploitation	116,103	115,162.87	511.21	2.18
BDE—Vanilla Exploration	116,004	115,517.26	238.08	1.88
BDE—Binary-Simple	117,018	116,215.94	427.61	1.28
BDE—Binary-Complex	116,786	116,180.45	437.48	1.31

Table A15. Solution quality on *mnknapcb6* (inst. 15, $n = 500$, $m = 10$, $f^* = 215,013$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	201,420	199,877.55	674.24	7.04
BPSO—Vanilla Exploration	205,458	203,911.74	547.92	5.16
BPSO—Binary-Simple	205,304	203,506.16	775.48	5.35
BPSO—Binary-Complex	204,812	203,355.26	652.27	5.42
GA—Vanilla Exploitation	214,726	214,494.39	145.63	0.24
GA—Vanilla Exploration	204,862	203,823.68	661.68	5.20
GA—Binary-Simple	214,261	212,154.42	2183.47	1.33
GA—Binary-Complex	214,501	214,143.48	225.19	0.40
BGWO—Vanilla Exploitation	206,007	204,888.77	428.65	4.71
BGWO—Vanilla Exploration	206,605	204,380.32	644.38	4.95
BGWO—Binary-Simple	205,625	204,548.74	446.96	4.87
BGWO—Binary-Complex	205,932	204,552.94	571.30	4.86
BDE—Vanilla Exploitation	207,447	205,470.45	1151.32	4.44
BDE—Vanilla Exploration	212,122	210,094.39	1099.69	2.29
BDE—Binary-Simple	212,487	211,499.35	854.60	1.63
BDE—Binary-Complex	212,827	211,311.61	965.79	1.72

Table A16. Solution quality on *mnknapcb6* (inst. 29, $n = 500$, $m = 10$, $f^* = 307,014$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	304,012	303,287.06	278.82	1.21
BPSO—Vanilla Exploration	304,803	304,274.65	209.11	0.89
BPSO—Binary-Simple	304,416	304,156.39	177.58	0.93
BPSO—Binary-Complex	304,579	304,123.16	175.84	0.94
GA—Vanilla Exploitation	306,762	306,636.00	80.01	0.12
GA—Vanilla Exploration	304,231	303,708.26	294.22	1.08
GA—Binary-Simple	306,408	305,403.32	759.77	0.52
GA—Binary-Complex	306,660	306,401.94	183.35	0.20
BGWO—Vanilla Exploitation	299,685	297,936.84	892.47	2.96
BGWO—Vanilla Exploration	299,685	298,085.06	737.99	2.91
BGWO—Binary-Simple	299,685	298,040.32	804.64	2.92
BGWO—Binary-Complex	299,685	298,069.61	759.80	2.91

Table A16. Cont.

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BDE—Vanilla Exploitation	305,610	304,930.48	433.79	0.68
BDE—Vanilla Exploration	305,887	305,541.23	185.06	0.48
BDE—Binary-Simple	306,324	305,955.00	190.80	0.34
BDE—Binary-Complex	306,352	305,925.94	210.04	0.35

Appendix A.7. Benchmark Group *mknapcb7*Table A17. Solution quality on mknapcb7 (inst. 0, $n = 100$, $m = 30$, $f^* = 21,946$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	21,946	21,722.74	84.37	1.02
BPSO—Vanilla Exploration	21,946	21,796.81	101.26	0.68
BPSO—Binary-Simple	21,840	21,730.16	65.37	0.98
BPSO—Binary-Complex	21,946	21,771.13	89.79	0.80
GA—Vanilla Exploitation	21,787	21,609.06	75.62	1.54
GA—Vanilla Exploration	21,946	21,768.84	76.96	0.81
GA—Binary-Simple	21,946	21,756.48	79.81	0.86
GA—Binary-Complex	21,946	21,756.90	76.52	0.86
BGWO—Vanilla Exploitation	21,787	21,660.00	45.05	1.30
BGWO—Vanilla Exploration	21,682	21,650.71	35.33	1.35
BGWO—Binary-Simple	21,787	21,648.97	52.08	1.35
BGWO—Binary-Complex	21,787	21,656.55	42.30	1.32
BDE—Vanilla Exploitation	21,772	21,619.39	59.86	1.49
BDE—Vanilla Exploration	21,787	21,729.84	47.83	0.98
BDE—Binary-Simple	21,874	21,711.23	55.91	1.07
BDE—Binary-Complex	21,946	21,712.61	64.68	1.06

Table A18. Solution quality on mknapcb7 (inst. 15, $n = 100$, $m = 30$, $f^* = 41,058$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	40,249	39,968.16	147.87	2.65
BPSO—Vanilla Exploration	40,813	40,451.39	101.87	1.48
BPSO—Binary-Simple	40,742	40,406.94	113.48	1.59
BPSO—Binary-Complex	40,840	40,403.94	166.20	1.59
GA—Vanilla Exploitation	40,954	40,783.97	104.79	0.67
GA—Vanilla Exploration	40,550	40,392.61	107.00	1.62
GA—Binary-Simple	40,912	40,645.00	175.47	1.01
GA—Binary-Complex	40,834	40,608.42	164.51	1.09
BGWO—Vanilla Exploitation	40,016	39,776.06	132.17	3.12
BGWO—Vanilla Exploration	40,006	39,684.26	153.00	3.35
BGWO—Binary-Simple	39,952	39,697.19	136.36	3.31
BGWO—Binary-Complex	39,990	39,667.19	129.03	3.39
BDE—Vanilla Exploitation	40,723	40,417.58	190.67	1.56
BDE—Vanilla Exploration	40,912	40,700.58	143.41	0.87
BDE—Binary-Simple	40,912	40,715.68	113.34	0.83
BDE—Binary-Complex	40,925	40,761.58	115.02	0.72

Table A19. Solution quality on mknapcb7 (inst. 29, $n = 100$, $m = 30$, $f^* = 60,603$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	60,414	60,270.97	80.42	0.55
BPSO—Vanilla Exploration	60,533	60,429.52	54.19	0.29
BPSO—Binary-Simple	60,533	60,415.81	50.91	0.31
BPSO—Binary-Complex	60,533	60,431.26	56.75	0.28

Table A19. Cont.

MH—Variant	Best Fitness	Average	Std.	Gap (%)
GA—Vanilla Exploitation	60,508	60,354.71	90.81	0.41
GA—Vanilla Exploration	60,508	60,341.26	97.09	0.43
GA—Binary-Simple	60,508	60,369.16	75.76	0.39
GA—Binary-Complex	60,508	60,362.45	70.99	0.40
BGWO—Vanilla Exploitation	59,568	59,192.48	145.81	2.33
BGWO—Vanilla Exploration	59,800	59,237.10	194.78	2.25
BGWO—Binary-Simple	59,464	59,173.77	149.93	2.36
BGWO—Binary-Complex	59,800	59,251.58	205.40	2.23
BDE—Vanilla Exploitation	60,533	60,339.23	102.85	0.44
BDE—Vanilla Exploration	60,533	60,477.10	44.35	0.21
BDE—Binary-Simple	60,533	60,489.45	38.19	0.19
BDE—Binary-Complex	60,533	60,475.77	45.03	0.21

Appendix A.8. Benchmark Group *mnknapcb8*Table A20. Solution quality on *mnknapcb8* (inst. 0, $n = 250$, $m = 30$, $f^* = 56,693$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	55,670	55,263.52	136.79	2.52
BPSO—Vanilla Exploration	55,916	55,641.94	126.98	1.85
BPSO—Binary-Simple	55,781	55,565.29	115.79	1.99
BPSO—Binary-Complex	55,973	55,615.55	174.69	1.90
GA—Vanilla Exploitation	56,423	56,139.06	141.06	0.98
GA—Vanilla Exploration	55,915	55,612.58	168.68	1.91
GA—Binary-Simple	56,407	55,814.03	239.39	1.55
GA—Binary-Complex	56,217	55,982.94	144.62	1.25
BGWO—Vanilla Exploitation	55,400	55,103.97	118.26	2.80
BGWO—Vanilla Exploration	55,460	55,078.39	129.34	2.85
BGWO—Binary-Simple	55,218	55,040.58	93.90	2.91
BGWO—Binary-Complex	55,460	55,109.03	141.23	2.79
BDE—Vanilla Exploitation	56,109	55,782.52	187.09	1.61
BDE—Vanilla Exploration	56,110	55,823.10	149.52	1.53
BDE—Binary-Simple	56,440	56,044.26	176.34	1.14
BDE—Binary-Complex	56,305	55,980.90	161.97	1.26

Table A21. Solution quality on *mnknapcb8* (inst. 15, $n = 250$, $m = 30$, $f^* = 107,246$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	102,880	102,137.61	262.54	4.76
BPSO—Vanilla Exploration	104,541	103,739.87	247.21	3.27
BPSO—Binary-Simple	104,192	103,605.94	226.54	3.39
BPSO—Binary-Complex	104,159	103,620.48	279.37	3.38
GA—Vanilla Exploitation	106,934	106,602.03	138.89	0.60
GA—Vanilla Exploration	104,486	103,703.52	338.62	3.30
GA—Binary-Simple	106,131	105,140.61	473.65	1.96
GA—Binary-Complex	106,657	106,355.84	197.22	0.83
BGWO—Vanilla Exploitation	103,269	102,876.97	219.55	4.07
BGWO—Vanilla Exploration	103,447	102,770.00	296.97	4.17
BGWO—Binary-Simple	103,417	102,824.06	274.93	4.12
BGWO—Binary-Complex	103,654	102,879.39	273.08	4.07
BDE—Vanilla Exploitation	105,228	104,170.32	663.05	2.87
BDE—Vanilla Exploration	106,091	105,268.71	298.59	1.84
BDE—Binary-Simple	106,202	105,609.87	315.00	1.53
BDE—Binary-Complex	106,259	105,647.29	252.66	1.49

Table A22. Solution quality on mknapcb8 (inst. 29, $n = 250$, $m = 30$, $f^* = 149,572$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	148,259	147,910.97	172.70	1.11
BPSO—Vanilla Exploration	148,666	148,428.03	133.23	0.76
BPSO—Binary-Simple	148,812	148,383.58	166.99	0.79
BPSO—Binary-Complex	148,696	148,383.77	148.43	0.79
GA—Vanilla Exploitation	149,284	149,112.23	128.48	0.31
GA—Vanilla Exploration	148,767	148,169.65	229.95	0.94
GA—Binary-Simple	149,261	148,497.55	310.52	0.72
GA—Binary-Complex	149,137	148,814.16	262.13	0.51
BGWO—Vanilla Exploitation	145,945	145,127.97	314.18	2.97
BGWO—Vanilla Exploration	145,945	145,327.90	273.81	2.84
BGWO—Binary-Simple	145,945	145,240.10	295.64	2.90
BGWO—Binary-Complex	145,945	145,303.03	287.58	2.85
BDE—Vanilla Exploitation	149,054	148,572.35	250.20	0.67
BDE—Vanilla Exploration	149,070	148,869.32	119.11	0.47
BDE—Binary-Simple	149,218	148,985.84	134.30	0.39
BDE—Binary-Complex	149,271	148,987.81	157.04	0.39

*Appendix A.9. Benchmark Group mknapcb9***Table A23.** Solution quality on mknapcb9 (inst. 15, $n = 500$, $m = 30$, $f^* = 215,762$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	201,843	200,709.52	647.97	6.98
BPSO—Vanilla Exploration	205,451	204,132.68	641.15	5.39
BPSO—Binary-Simple	204,784	203,873.71	441.56	5.51
BPSO—Binary-Complex	204,890	203,637.90	584.08	5.62
GA—Vanilla Exploitation	215,101	214,781.45	181.33	0.45
GA—Vanilla Exploration	204,940	203,765.06	664.82	5.56
GA—Binary-Simple	214,929	212,189.71	2543.54	1.66
GA—Binary-Complex	214,741	214,209.94	395.18	0.72
BGWO—Vanilla Exploitation	205,869	204,880.65	443.83	5.04
BGWO—Vanilla Exploration	205,981	204,306.42	442.42	5.31
BGWO—Binary-Simple	205,758	204,364.74	507.83	5.28
BGWO—Binary-Complex	205,483	204,369.65	414.54	5.28
BDE—Vanilla Exploitation	209,597	205,982.87	1535.50	4.53
BDE—Vanilla Exploration	210,659	209,327.00	692.37	2.98
BDE—Binary-Simple	212,450	211,331.74	624.61	2.05
BDE—Binary-Complex	212,826	210,918.19	1043.15	2.24

Table A24. Solution quality on mknapcb9 (inst. 29, $n = 500$, $m = 30$, $f^* = 300,460$).

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BPSO—Vanilla Exploitation	296,383	295,862.55	291.46	1.53
BPSO—Vanilla Exploration	297,374	296,976.94	197.75	1.16
BPSO—Binary-Simple	297,331	296,850.26	226.18	1.20
BPSO—Binary-Complex	297,608	296,881.26	286.56	1.19
GA—Vanilla Exploitation	300,150	299,914.39	186.78	0.18
GA—Vanilla Exploration	297,184	296,304.97	317.56	1.38
GA—Binary-Simple	300,056	298,440.19	1002.70	0.67
GA—Binary-Complex	300,017	299,666.77	237.73	0.26
BGWO—Vanilla Exploitation	291,564	290,310.81	753.75	3.38
BGWO—Vanilla Exploration	291,564	290,453.00	652.20	3.33
BGWO—Binary-Simple	291,564	290,403.55	671.79	3.35
BGWO—Binary-Complex	291,564	290,471.94	641.31	3.32

Table A24. Cont.

MH—Variant	Best Fitness	Average	Std.	Gap (%)
BDE—Vanilla Exploitation	298,666	297,757.39	485.18	0.90
BDE—Vanilla Exploration	298,899	298,359.13	274.61	0.70
BDE—Binary-Simple	299,503	298,739.42	347.99	0.57
BDE—Binary-Complex	299,389	298,765.19	299.12	0.56

Appendix B. Complete Pairwise Statistical Results

The following table reports all 108 condition-level Wilcoxon comparisons, with 31 fitness observations paired by stored run order. The displayed p -values are unadjusted two-sided values. A + or − denotes an unadjusted $p < 0.05$ with a positive or negative mean fitness difference for Binary-Complex, respectively; $\approx$ denotes no rejection at that threshold, not equivalence. These 324 comparisons are exploratory. Bonferroni-Dunn-adjusted global rank comparisons are reported separately in Table 11.

Table A25. Unadjusted two-sided Wilcoxon results for all 27 selected instances and four metaheuristics ($R = 31$).

Instance	Algorithm	vs. V-Exploit		vs. V-Explore		vs. Bin-Simple	
		p	Sign	p	Sign	p	Sign
mknapcb1[0]	BPSO	5.726×10^{-6}	+	0.1610	$\approx$	0.8373	$\approx$
mknapcb1[0]	GA	0.0161	+	0.2219	$\approx$	0.1261	$\approx$
mknapcb1[0]	BGWO	0.3719	$\approx$	0.7220	$\approx$	0.1846	$\approx$
mknapcb1[0]	BDE	0.0036	+	5.621×10^{-5}	−	0.0875	$\approx$
mknapcb1[15]	BPSO	1.174×10^{-6}	+	0.9483	$\approx$	0.3388	$\approx$
mknapcb1[15]	GA	0.7464	$\approx$	6.673×10^{-5}	+	0.0461	+
mknapcb1[15]	BGWO	0.7775	$\approx$	0.0021	+	0.2311	$\approx$
mknapcb1[15]	BDE	1.025×10^{-5}	+	0.8129	$\approx$	0.5429	$\approx$
mknapcb1[29]	BPSO	7.479×10^{-5}	+	0.6441	$\approx$	0.4222	$\approx$
mknapcb1[29]	GA	0.0760	$\approx$	0.0016	+	0.8078	$\approx$
mknapcb1[29]	BGWO	0.2218	$\approx$	0.8647	$\approx$	0.9357	$\approx$
mknapcb1[29]	BDE	1.827×10^{-4}	+	0.5637	$\approx$	0.4142	$\approx$
mknapcb2[0]	BPSO	9.313×10^{-10}	+	0.0699	$\approx$	0.6883	$\approx$
mknapcb2[0]	GA	1.494×10^{-4}	−	1.173×10^{-6}	+	0.0063	+
mknapcb2[0]	BGWO	0.0571	$\approx$	0.8303	$\approx$	0.2473	$\approx$
mknapcb2[0]	BDE	1.863×10^{-9}	+	2.882×10^{-4}	+	0.3777	$\approx$
mknapcb2[15]	BPSO	9.313×10^{-10}	+	0.0829	$\approx$	0.7498	$\approx$
mknapcb2[15]	GA	4.729×10^{-6}	−	1.734×10^{-6}	+	3.464×10^{-6}	+
mknapcb2[15]	BGWO	0.0282	−	0.5922	$\approx$	0.7213	$\approx$
mknapcb2[15]	BDE	1.863×10^{-9}	+	0.1959	$\approx$	0.0977	$\approx$
mknapcb2[29]	BPSO	1.577×10^{-6}	+	0.0015	−	0.7023	$\approx$
mknapcb2[29]	GA	9.929×10^{-4}	−	9.313×10^{-10}	+	0.0035	+
mknapcb2[29]	BGWO	0.2012	$\approx$	0.3454	$\approx$	0.9165	$\approx$
mknapcb2[29]	BDE	1.577×10^{-6}	+	0.2363	$\approx$	0.4159	$\approx$
mknapcb3[0]	BPSO	9.313×10^{-10}	+	0.0168	−	0.5168	$\approx$
mknapcb3[0]	GA	6.941×10^{-5}	−	9.313×10^{-10}	+	1.863×10^{-9}	+
mknapcb3[0]	BGWO	0.1795	$\approx$	0.2548	$\approx$	0.5481	$\approx$
mknapcb3[0]	BDE	9.313×10^{-10}	+	4.005×10^{-8}	+	0.8092	$\approx$
mknapcb3[15]	BPSO	9.313×10^{-10}	+	5.906×10^{-4}	−	0.5043	$\approx$
mknapcb3[15]	GA	6.565×10^{-6}	−	9.313×10^{-10}	+	5.960×10^{-7}	+
mknapcb3[15]	BGWO	0.0132	−	0.6938	$\approx$	0.1011	$\approx$
mknapcb3[15]	BDE	9.313×10^{-10}	+	0.8393	$\approx$	0.7688	$\approx$
mknapcb3[29]	BPSO	1.174×10^{-6}	+	0.0079	−	0.9687	$\approx$
mknapcb3[29]	GA	3.927×10^{-6}	−	1.173×10^{-6}	+	2.859×10^{-7}	+
mknapcb3[29]	BGWO	0.1386	$\approx$	0.8927	$\approx$	0.7150	$\approx$
mknapcb3[29]	BDE	9.313×10^{-10}	+	3.407×10^{-6}	+	0.0977	$\approx$

Table A25. Cont.

Instance	Algorithm	vs. V-Exploit p	Sign	vs. V-Explore p	Sign	vs. Bin-Simple p	Sign
mknapcb4 [0]	BPSO	1.131×10^{-4}	+	0.1474	≈	0.1554	≈
mknapcb4 [0]	GA	0.1442	≈	0.6894	≈	0.9273	≈
mknapcb4 [0]	BGWO	0.8589	≈	0.7764	≈	0.5595	≈
mknapcb4 [0]	BDE	0.0039	+	0.6808	≈	0.2794	≈
mknapcb4 [15]	BPSO	1.173×10^{-6}	+	0.5508	≈	0.9609	≈
mknapcb4 [15]	GA	0.0179	−	0.0045	+	0.5521	≈
mknapcb4 [15]	BGWO	0.3745	≈	0.3754	≈	0.3331	≈
mknapcb4 [15]	BDE	3.880×10^{-6}	+	0.5999	≈	0.1945	≈
mknapcb4 [29]	BPSO	6.519×10^{-8}	+	0.0037	−	0.6146	≈
mknapcb4 [29]	GA	0.0332	+	0.5345	≈	0.8082	≈
mknapcb4 [29]	BGWO	0.0256	+	0.5016	≈	0.5161	≈
mknapcb4 [29]	BDE	4.975×10^{-5}	+	0.0161	−	0.5308	≈
mknapcb5 [0]	BPSO	2.111×10^{-6}	+	7.148×10^{-4}	−	0.5566	≈
mknapcb5 [0]	GA	6.047×10^{-4}	−	1.174×10^{-6}	+	0.2396	≈
mknapcb5 [0]	BGWO	0.0192	−	0.8552	≈	0.1763	≈
mknapcb5 [0]	BDE	2.818×10^{-6}	+	0.0024	+	0.0187	+
mknapcb5 [15]	BPSO	9.313×10^{-10}	+	0.5832	≈	0.3570	≈
mknapcb5 [15]	GA	3.464×10^{-6}	−	1.174×10^{-6}	+	8.196×10^{-8}	+
mknapcb5 [15]	BGWO	0.2319	≈	0.3313	≈	0.0020	+
mknapcb5 [15]	BDE	9.313×10^{-10}	+	1.276×10^{-7}	+	0.7316	≈
mknapcb5 [29]	BPSO	1.174×10^{-6}	+	0.4590	≈	0.5370	≈
mknapcb5 [29]	GA	2.825×10^{-5}	−	2.563×10^{-6}	+	0.2170	≈
mknapcb5 [29]	BGWO	0.0029	+	0.5525	≈	0.1961	≈
mknapcb5 [29]	BDE	1.770×10^{-8}	+	2.207×10^{-5}	+	0.5632	≈
mknapcb6 [0]	BPSO	1.863×10^{-9}	+	0.0037	−	0.8141	≈
mknapcb6 [0]	GA	6.562×10^{-6}	−	1.733×10^{-6}	+	1.024×10^{-7}	+
mknapcb6 [0]	BGWO	7.017×10^{-4}	−	0.0875	≈	0.3369	≈
mknapcb6 [0]	BDE	4.742×10^{-6}	+	5.960×10^{-7}	+	0.8754	≈
mknapcb6 [15]	BPSO	9.313×10^{-10}	+	0.0036	−	0.3673	≈
mknapcb6 [15]	GA	3.746×10^{-6}	−	1.174×10^{-6}	+	9.313×10^{-10}	+
mknapcb6 [15]	BGWO	0.0064	−	0.0355	+	0.6938	≈
mknapcb6 [15]	BDE	9.313×10^{-10}	+	1.554×10^{-4}	+	0.1574	≈
mknapcb6 [29]	BPSO	1.863×10^{-9}	+	0.0351	−	0.3223	≈
mknapcb6 [29]	GA	5.454×10^{-6}	−	9.313×10^{-10}	+	4.163×10^{-7}	+
mknapcb6 [29]	BGWO	0.0117	+	0.6547	≈	0.1088	≈
mknapcb6 [29]	BDE	1.295×10^{-6}	+	1.825×10^{-6}	+	0.7207	≈
mknapcb7 [0]	BPSO	0.0344	+	0.1471	≈	0.0157	+
mknapcb7 [0]	GA	1.866×10^{-6}	+	0.3008	≈	1.0000	≈
mknapcb7 [0]	BGWO	0.7930	≈	0.5527	≈	0.2076	≈
mknapcb7 [0]	BDE	1.790×10^{-5}	+	0.1918	≈	0.8012	≈
mknapcb7 [15]	BPSO	1.863×10^{-9}	+	0.0479	−	0.8217	≈
mknapcb7 [15]	GA	1.771×10^{-4}	−	2.148×10^{-5}	+	0.4688	≈
mknapcb7 [15]	BGWO	0.0048	−	0.7164	≈	0.2850	≈
mknapcb7 [15]	BDE	1.738×10^{-6}	+	0.1086	≈	0.0828	≈
mknapcb7 [29]	BPSO	1.304×10^{-8}	+	0.7533	≈	0.3388	≈
mknapcb7 [29]	GA	0.7464	≈	0.3065	≈	0.7468	≈
mknapcb7 [29]	BGWO	0.1064	≈	0.7172	≈	0.1718	≈
mknapcb7 [29]	BDE	9.311×10^{-6}	+	0.9272	≈	0.3032	≈
mknapcb8 [0]	BPSO	1.863×10^{-9}	+	0.4389	≈	0.2133	≈
mknapcb8 [0]	GA	6.499×10^{-4}	−	1.733×10^{-6}	+	0.0052	+
mknapcb8 [0]	BGWO	0.9154	≈	0.2586	≈	0.0267	+
mknapcb8 [0]	BDE	1.968×10^{-5}	+	0.0014	+	0.1701	≈
mknapcb8 [15]	BPSO	9.313×10^{-10}	+	0.0407	−	0.9344	≈
mknapcb8 [15]	GA	1.895×10^{-6}	−	9.313×10^{-10}	+	9.313×10^{-10}	+
mknapcb8 [15]	BGWO	0.7913	≈	0.0416	+	0.3158	≈
mknapcb8 [15]	BDE	1.173×10^{-6}	+	8.020×10^{-6}	+	0.5948	≈
mknapcb8 [29]	BPSO	4.657×10^{-9}	+	0.1086	≈	0.8092	≈
mknapcb8 [29]	GA	1.928×10^{-7}	−	2.603×10^{-6}	+	1.742×10^{-4}	+
mknapcb8 [29]	BGWO	0.0148	+	0.6378	≈	0.1024	≈
mknapcb8 [29]	BDE	9.313×10^{-10}	+	0.0019	+	0.8242	≈

Table A25. Cont.

Instance	Algorithm	vs. V-Exploit p	Sign	vs. V-Explore p	Sign	vs. Bin-Simple p	Sign
mknapcb9 [0]	BPSO	9.313×10^{-10}	+	0.9397	$\approx$	0.8677	$\approx$
mknapcb9 [0]	GA	9.313×10^{-10}	—	9.313×10^{-10}	+	7.195×10^{-6}	+
mknapcb9 [0]	BGWO	0.5115	$\approx$	0.7509	$\approx$	0.2724	$\approx$
mknapcb9 [0]	BDE	2.560×10^{-6}	+	1.276×10^{-7}	+	0.9001	$\approx$
mknapcb9 [15]	BPSO	9.313×10^{-10}	+	0.0027	—	0.1461	$\approx$
mknapcb9 [15]	GA	4.657×10^{-9}	—	1.174×10^{-6}	+	5.613×10^{-5}	+
mknapcb9 [15]	BGWO	6.987×10^{-5}	—	0.5057	$\approx$	0.9818	$\approx$
mknapcb9 [15]	BDE	9.313×10^{-10}	+	7.097×10^{-7}	+	0.1574	$\approx$
mknapcb9 [29]	BPSO	9.313×10^{-10}	+	0.1302	$\approx$	0.6435	$\approx$
mknapcb9 [29]	GA	7.539×10^{-5}	—	1.174×10^{-6}	+	2.356×10^{-7}	+
mknapcb9 [29]	BGWO	0.0117	+	0.1797	$\approx$	0.1088	$\approx$
mknapcb9 [29]	BDE	1.738×10^{-6}	+	2.627×10^{-5}	+	0.8242	$\approx$
Total (+ / $\approx$ / —)		62 / 19 / 27		40 / 54 / 14		21 / 87 / 0	

Illustrative Convergence Trajectories

These plots extend the trajectory illustrations to mknapcb5 and mknapcb9. For each variant, the displayed run is selected separately for each metaheuristic and instance as that condition's best-final-fitness run among 31 independent runs; indices 0, 15, and 29 appear from top to bottom. The plots illustrate trajectory progression only, not means or typical behavior, and support no causal or superiority conclusion.

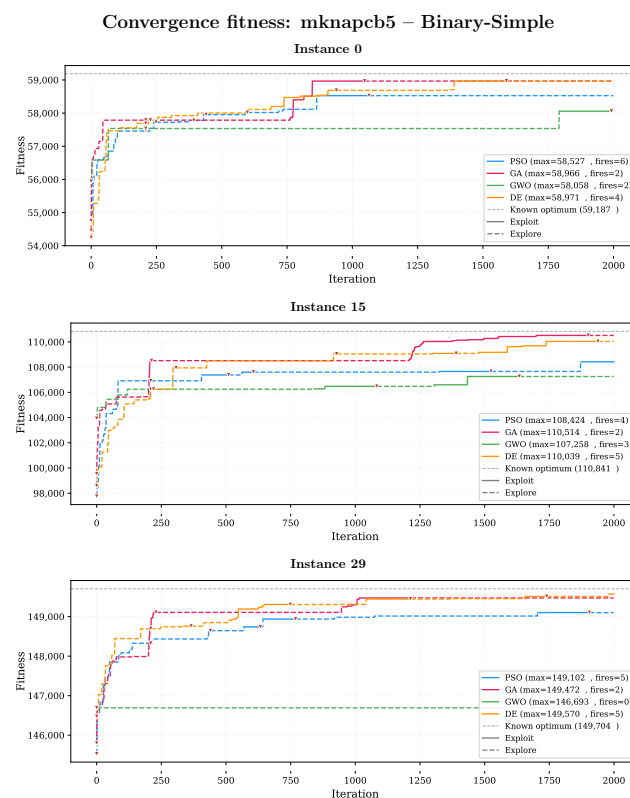


Figure A1. Selected Binary-Simple convergence trajectories for mknapcb5; instances 0, 15, and 29 appear from top to bottom. For each metaheuristic and instance, the curve is the best-final-fitness run among 31 independent runs.

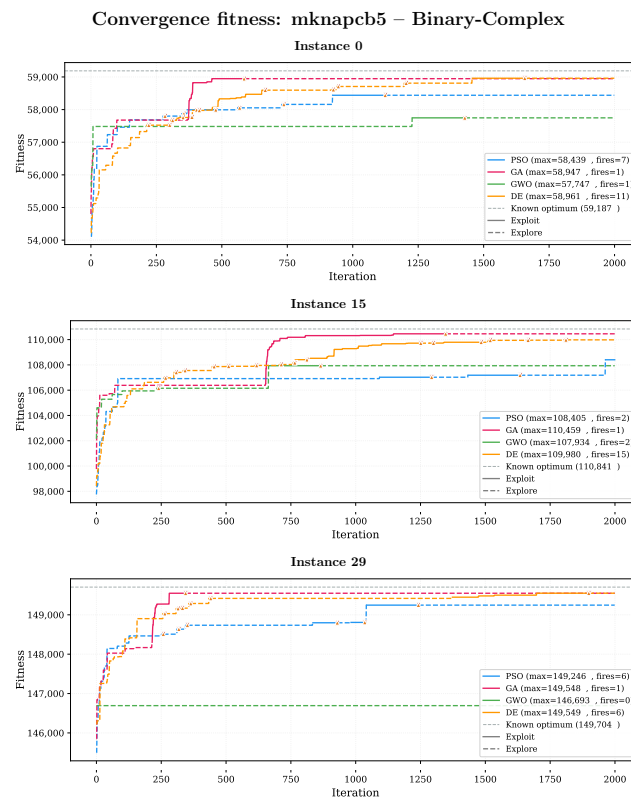


Figure A2. Selected Binary-Complex convergence trajectories for mknapcb5; instances 0, 15, and 29 appear from top to bottom. For each metaheuristic and instance, the curve is the best-final-fitness run among 31 independent runs.

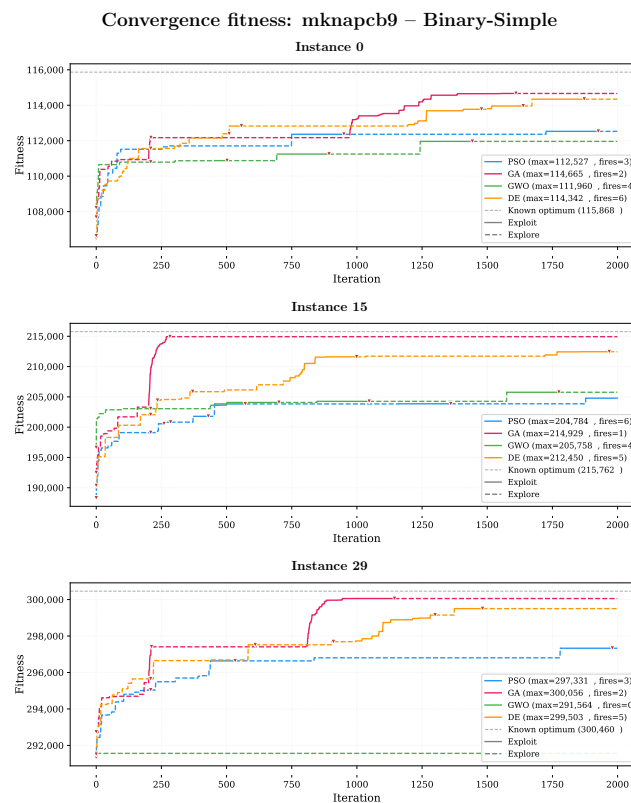


Figure A3. Selected Binary-Simple convergence trajectories for mknapcb9; instances 0, 15, and 29 appear from top to bottom. For each metaheuristic and instance, the curve is the best-final-fitness run among 31 independent runs.

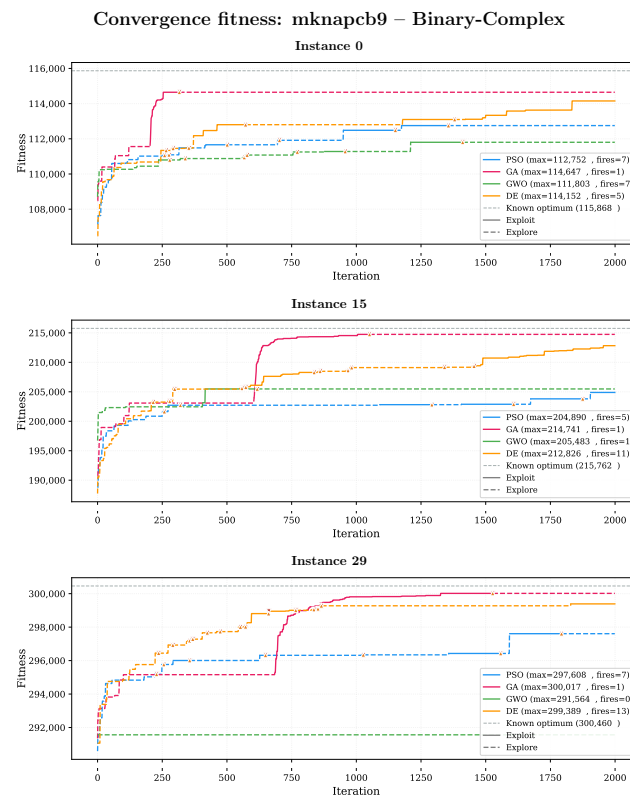


Figure A4. Selected Binary-Complex convergence trajectories for mknapcb9; instances 0, 15, and 29 appear from top to bottom. For each metaheuristic and instance, the curve is the best-final-fitness run among 31 independent runs.

Appendix C. Complete Computational Runtime Results

Execution time in seconds is reported as mean $\pm$ sample standard deviation over the 31 independent runs in each condition. These are per-instance runtime summaries, not pooled estimates across different problem sizes.

Table A26. Execution time [s] (mean $\pm$ sample standard deviation) for all 27 selected MKP instances ($R = 31$).

Instance	Algorithm	V-Exploit	V-Explore	Bin-Simple	Bin-Complex
mknapcb1[0]	BPSO	43.6 $\pm$ 0.3	43.8 $\pm$ 0.5	49.2 $\pm$ 1.0	50.0 $\pm$ 1.2
mknapcb1[0]	GA	29.1 $\pm$ 0.9	38.4 $\pm$ 1.1	43.1 $\pm$ 1.4	43.3 $\pm$ 1.4
mknapcb1[0]	BGWO	47.7 $\pm$ 0.8	47.6 $\pm$ 0.4	53.7 $\pm$ 1.1	53.3 $\pm$ 1.1
mknapcb1[0]	BDE	46.4 $\pm$ 1.2	46.5 $\pm$ 1.0	52.8 $\pm$ 1.6	49.2 $\pm$ 2.8
mknapcb1[15]	BPSO	23.6 $\pm$ 0.4	24.3 $\pm$ 0.2	29.2 $\pm$ 0.3	29.9 $\pm$ 1.1
mknapcb1[15]	GA	19.1 $\pm$ 0.3	21.4 $\pm$ 0.4	26.4 $\pm$ 0.7	26.6 $\pm$ 1.3
mknapcb1[15]	BGWO	29.4 $\pm$ 0.4	29.1 $\pm$ 0.3	34.9 $\pm$ 1.1	34.9 $\pm$ 1.6
mknapcb1[15]	BDE	31.9 $\pm$ 2.0	31.3 $\pm$ 1.2	38.0 $\pm$ 1.2	35.2 $\pm$ 1.4
mknapcb1[29]	BPSO	18.2 $\pm$ 0.2	17.7 $\pm$ 0.2	23.1 $\pm$ 0.3	23.5 $\pm$ 0.7
mknapcb1[29]	GA	10.8 $\pm$ 0.3	12.3 $\pm$ 0.3	17.6 $\pm$ 0.6	17.4 $\pm$ 0.5
mknapcb1[29]	BGWO	14.9 $\pm$ 0.2	15.0 $\pm$ 0.2	20.4 $\pm$ 0.4	20.6 $\pm$ 0.7
mknapcb1[29]	BDE	23.0 $\pm$ 0.8	22.6 $\pm$ 0.4	28.4 $\pm$ 1.0	26.7 $\pm$ 0.6
mknapcb2[0]	BPSO	110.1 $\pm$ 1.0	111.1 $\pm$ 1.3	117.4 $\pm$ 1.9	119.0 $\pm$ 2.7
mknapcb2[0]	GA	85.1 $\pm$ 1.3	97.5 $\pm$ 1.9	103.4 $\pm$ 2.4	102.9 $\pm$ 2.4
mknapcb2[0]	BGWO	119.2 $\pm$ 1.3	118.2 $\pm$ 1.8	125.7 $\pm$ 2.6	126.3 $\pm$ 3.5
mknapcb2[0]	BDE	118.9 $\pm$ 1.9	118.9 $\pm$ 1.7	126.3 $\pm$ 3.1	117.0 $\pm$ 4.8
mknapcb2[15]	BPSO	52.4 $\pm$ 0.7	54.5 $\pm$ 0.8	59.7 $\pm$ 0.8	60.5 $\pm$ 1.4
mknapcb2[15]	GA	49.5 $\pm$ 1.3	47.1 $\pm$ 0.8	53.1 $\pm$ 1.2	53.8 $\pm$ 1.6
mknapcb2[15]	BGWO	68.4 $\pm$ 0.6	67.8 $\pm$ 0.8	75.1 $\pm$ 1.7	74.9 $\pm$ 1.6
mknapcb2[15]	BDE	65.9 $\pm$ 5.8	67.7 $\pm$ 2.9	78.0 $\pm$ 4.5	72.4 $\pm$ 4.2

Table A26. Cont.

Instance	Algorithm	V-Exploit	V-Explore	Bin-Simple	Bin-Complex
mknapcb2[29]	BPSO	44.8 ± 0.7	44.0 ± 0.4	49.6 ± 0.7	50.9 ± 1.5
mknapcb2[29]	GA	24.7 ± 0.6	27.6 ± 0.7	33.1 ± 0.8	33.1 ± 1.0
mknapcb2[29]	BGWO	32.6 ± 0.3	33.2 ± 0.4	39.2 ± 0.9	38.9 ± 0.9
mknapcb2[29]	BDE	54.0 ± 2.0	53.8 ± 1.1	60.5 ± 1.5	56.9 ± 2.2
mknapcb3[0]	BPSO	219.1 ± 2.1	220.6 ± 3.2	230.0 ± 3.9	232.0 ± 4.5
mknapcb3[0]	GA	187.7 ± 3.3	199.6 ± 11.2	202.9 ± 3.1	203.1 ± 4.3
mknapcb3[0]	BGWO	237.3 ± 5.2	236.5 ± 7.7	245.4 ± 4.6	244.5 ± 5.6
mknapcb3[0]	BDE	234.8 ± 5.8	234.9 ± 5.9	246.2 ± 4.8	225.1 ± 16.0
mknapcb3[15]	BPSO	107.3 ± 1.1	109.6 ± 0.9	115.4 ± 1.9	117.1 ± 1.9
mknapcb3[15]	GA	112.9 ± 2.4	96.3 ± 2.1	103.3 ± 2.6	105.4 ± 2.5
mknapcb3[15]	BGWO	139.9 ± 1.4	138.1 ± 2.1	146.9 ± 2.8	147.6 ± 3.6
mknapcb3[15]	BDE	138.8 ± 8.0	140.3 ± 6.2	156.0 ± 9.3	144.9 ± 10.2
mknapcb3[29]	BPSO	93.0 ± 1.1	91.4 ± 1.0	97.4 ± 1.3	99.0 ± 2.4
mknapcb3[29]	GA	49.4 ± 1.1	56.5 ± 1.2	61.8 ± 1.1	61.8 ± 1.8
mknapcb3[29]	BGWO	65.3 ± 0.6	66.4 ± 0.9	73.4 ± 1.5	73.5 ± 2.0
mknapcb3[29]	BDE	106.9 ± 3.6	107.3 ± 2.2	115.0 ± 2.5	108.7 ± 3.5
mknapcb4[0]	BPSO	45.6 ± 3.2	45.4 ± 2.5	50.9 ± 1.4	51.0 ± 3.1
mknapcb4[0]	GA	29.2 ± 0.2	38.9 ± 0.2	43.7 ± 3.0	43.0 ± 1.0
mknapcb4[0]	BGWO	48.8 ± 1.8	49.0 ± 3.2	54.0 ± 0.4	54.2 ± 2.2
mknapcb4[0]	BDE	47.4 ± 2.2	47.0 ± 0.4	52.5 ± 3.2	49.6 ± 2.1
mknapcb4[15]	BPSO	25.1 ± 1.6	25.9 ± 1.6	30.8 ± 0.2	30.8 ± 0.1
mknapcb4[15]	GA	19.8 ± 0.2	22.7 ± 1.4	27.6 ± 1.5	27.4 ± 1.4
mknapcb4[15]	BGWO	30.8 ± 1.9	30.1 ± 0.1	35.6 ± 1.0	35.3 ± 0.2
mknapcb4[15]	BDE	31.6 ± 0.7	31.6 ± 2.1	37.0 ± 2.0	35.0 ± 1.4
mknapcb4[29]	BPSO	18.6 ± 1.3	17.8 ± 1.2	23.2 ± 1.3	23.0 ± 0.2
mknapcb4[29]	GA	10.9 ± 0.1	12.6 ± 0.1	17.8 ± 0.6	17.8 ± 1.0
mknapcb4[29]	BGWO	15.3 ± 1.0	15.5 ± 1.0	20.6 ± 0.5	20.4 ± 0.1
mknapcb4[29]	BDE	22.7 ± 0.5	22.6 ± 0.3	28.0 ± 1.8	26.9 ± 0.9
mknapcb5[0]	BPSO	116.7 ± 6.2	117.3 ± 7.1	121.8 ± 8.6	122.0 ± 5.4
mknapcb5[0]	GA	87.7 ± 0.5	101.0 ± 1.3	104.8 ± 0.8	104.4 ± 0.9
mknapcb5[0]	BGWO	125.9 ± 8.7	124.5 ± 8.0	128.0 ± 0.4	129.5 ± 7.0
mknapcb5[0]	BDE	121.9 ± 1.9	121.8 ± 4.9	127.9 ± 9.2	119.2 ± 3.6
mknapcb5[15]	BPSO	58.3 ± 4.2	59.8 ± 0.4	65.3 ± 4.3	65.2 ± 4.2
mknapcb5[15]	GA	54.7 ± 0.4	52.5 ± 3.7	57.0 ± 0.4	57.8 ± 3.8
mknapcb5[15]	BGWO	75.0 ± 2.6	73.2 ± 0.4	78.7 ± 1.7	78.2 ± 0.3
mknapcb5[15]	BDE	75.9 ± 5.2	76.0 ± 5.5	81.5 ± 2.0	78.0 ± 3.9
mknapcb5[29]	BPSO	46.0 ± 3.3	44.3 ± 0.2	50.0 ± 2.0	50.5 ± 3.5
mknapcb5[29]	GA	25.3 ± 0.4	28.9 ± 1.9	33.8 ± 1.3	33.5 ± 0.2
mknapcb5[29]	BGWO	33.6 ± 1.7	34.2 ± 2.3	39.6 ± 1.8	39.2 ± 0.5
mknapcb5[29]	BDE	54.2 ± 2.3	54.0 ± 0.5	59.2 ± 0.8	57.7 ± 2.7
mknapcb6[0]	BPSO	239.2 ± 3.6	238.4 ± 7.6	241.2 ± 0.5	244.4 ± 15.2
mknapcb6[0]	GA	200.4 ± 4.3	208.7 ± 5.1	215.4 ± 13.8	212.2 ± 0.7
mknapcb6[0]	BGWO	254.3 ± 6.3	251.5 ± 3.8	261.2 ± 16.4	260.9 ± 14.5
mknapcb6[0]	BDE	245.1 ± 6.2	246.7 ± 14.8	251.7 ± 1.6	236.2 ± 12.0
mknapcb6[15]	BPSO	112.2 ± 7.0	114.5 ± 0.6	119.1 ± 0.6	119.1 ± 0.5
mknapcb6[15]	GA	115.8 ± 0.6	99.7 ± 4.7	106.1 ± 6.2	107.7 ± 6.2
mknapcb6[15]	BGWO	147.4 ± 4.3	145.0 ± 8.7	149.8 ± 0.5	149.8 ± 0.5
mknapcb6[15]	BDE	144.0 ± 6.9	148.5 ± 11.1	161.1 ± 9.7	153.0 ± 12.7
mknapcb6[29]	BPSO	97.7 ± 3.9	95.0 ± 0.6	102.0 ± 6.2	101.3 ± 0.3
mknapcb6[29]	GA	53.3 ± 0.5	60.5 ± 3.7	65.7 ± 3.7	65.1 ± 3.8
mknapcb6[29]	BGWO	69.3 ± 2.7	70.6 ± 4.1	75.3 ± 0.3	75.9 ± 3.5
mknapcb6[29]	BDE	112.5 ± 6.0	111.1 ± 1.3	119.7 ± 8.0	112.3 ± 2.8
mknapcb7[0]	BPSO	49.4 ± 1.1	50.2 ± 3.6	55.1 ± 1.2	55.2 ± 1.2
mknapcb7[0]	GA	32.7 ± 0.4	43.8 ± 3.1	47.1 ± 1.3	47.4 ± 3.2
mknapcb7[0]	BGWO	53.5 ± 3.7	52.4 ± 0.3	58.6 ± 3.5	58.5 ± 3.6
mknapcb7[0]	BDE	53.6 ± 0.6	53.9 ± 4.1	58.7 ± 0.9	55.7 ± 1.6
mknapcb7[15]	BPSO	28.8 ± 1.8	29.9 ± 1.9	34.7 ± 1.9	34.5 ± 1.3
mknapcb7[15]	GA	22.3 ± 0.2	25.2 ± 0.1	30.5 ± 1.0	30.5 ± 1.7
mknapcb7[15]	BGWO	35.7 ± 2.2	35.2 ± 2.1	40.4 ± 1.2	40.1 ± 0.3
mknapcb7[15]	BDE	34.2 ± 1.2	34.3 ± 0.7	39.9 ± 1.0	38.4 ± 2.9

Table A26. Cont.

Instance	Algorithm	V-Exploit	V-Explore	Bin-Simple	Bin-Complex
mknapcb7 [29]	BPSO	18.8 ± 1.3	18.0 ± 1.2	23.2 ± 0.1	23.4 ± 1.2
mknapcb7 [29]	GA	12.0 ± 0.2	13.5 ± 0.8	18.5 ± 0.9	18.5 ± 0.9
mknapcb7 [29]	BGWO	15.9 ± 1.0	15.8 ± 0.1	21.2 ± 1.0	21.0 ± 0.1
mknapcb7 [29]	BDE	23.4 ± 0.6	23.5 ± 1.4	28.5 ± 0.3	27.6 ± 1.2
mknapcb8 [0]	BPSO	136.6 ± 4.0	135.9 ± 0.8	142.4 ± 8.3	142.8 ± 8.9
mknapcb8 [0]	GA	104.2 ± 0.6	119.8 ± 7.6	123.6 ± 2.5	122.3 ± 1.2
mknapcb8 [0]	BGWO	145.4 ± 9.2	142.3 ± 0.5	148.5 ± 0.7	150.0 ± 8.5
mknapcb8 [0]	BDE	143.4 ± 9.2	142.7 ± 8.9	148.7 ± 9.1	138.8 ± 4.8
mknapcb8 [15]	BPSO	68.3 ± 4.3	69.8 ± 0.4	75.1 ± 1.1	74.7 ± 0.3
mknapcb8 [15]	GA	63.6 ± 0.8	61.1 ± 4.1	66.0 ± 3.9	66.3 ± 4.0
mknapcb8 [15]	BGWO	87.2 ± 5.9	84.7 ± 0.4	90.6 ± 2.6	90.0 ± 0.3
mknapcb8 [15]	BDE	85.4 ± 2.3	86.8 ± 4.6	93.6 ± 5.7	89.4 ± 5.1
mknapcb8 [29]	BPSO	49.4 ± 2.0	48.1 ± 3.1	53.0 ± 1.4	52.8 ± 0.9
mknapcb8 [29]	GA	28.8 ± 0.4	32.1 ± 2.1	37.3 ± 2.0	36.7 ± 0.6
mknapcb8 [29]	BGWO	36.6 ± 2.2	36.8 ± 0.2	42.1 ± 1.2	42.2 ± 2.1
mknapcb8 [29]	BDE	57.4 ± 1.8	57.2 ± 1.6	63.4 ± 3.8	60.0 ± 1.6
mknapcb9 [0]	BPSO	288.8 ± 10.2	286.2 ± 1.1	293.2 ± 17.2	294.7 ± 17.4
mknapcb9 [0]	GA	240.6 ± 1.4	252.4 ± 14.7	255.4 ± 1.3	255.7 ± 2.6
mknapcb9 [0]	BGWO	304.3 ± 18.2	299.6 ± 1.1	307.1 ± 1.8	310.7 ± 15.9
mknapcb9 [0]	BDE	299.7 ± 18.1	299.5 ± 17.8	308.3 ± 18.8	286.8 ± 13.4
mknapcb9 [15]	BPSO	135.1 ± 3.6	139.3 ± 7.6	141.8 ± 0.6	143.3 ± 7.8
mknapcb9 [15]	GA	137.1 ± 1.6	118.4 ± 0.8	126.0 ± 6.7	126.2 ± 1.0
mknapcb9 [15]	BGWO	175.2 ± 9.7	171.7 ± 9.5	177.0 ± 1.8	178.4 ± 9.2
mknapcb9 [15]	BDE	168.9 ± 5.5	172.9 ± 10.7	185.9 ± 9.8	173.8 ± 5.6
mknapcb9 [29]	BPSO	109.6 ± 5.2	105.8 ± 6.2	110.4 ± 0.4	111.8 ± 6.2
mknapcb9 [29]	GA	62.1 ± 0.9	68.3 ± 3.8	73.2 ± 3.8	72.6 ± 3.9
mknapcb9 [29]	BGWO	77.4 ± 4.2	77.3 ± 0.3	83.4 ± 4.3	82.8 ± 0.6
mknapcb9 [29]	BDE	123.8 ± 3.6	123.6 ± 7.6	129.0 ± 1.8	125.9 ± 6.9

References

1. Dhiman, R.; Sood, M.; Thakur, J. A decade survey on exploration-exploitation trade-off in Metaheuristics: A proposal for conceptual framework-based validation model. *Comput. Sci. Rev.* **2026**, *61*, 100944.
2. Črepinšek, M.; Liu, S.H.; Mernik, M. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Comput. Surv. (CSUR)* **2013**, *45*, 35.
3. Zhang, H.; Sun, J.; Liu, T.; Zhang, K.; Zhang, Q. Balancing exploration and exploitation in multiobjective evolutionary optimization. *Inf. Sci.* **2019**, *497*, 129–148.
4. Vega, E.; Soto, R.; Crawford, B.; Peña, J.; Castro, C. A learning-based hybrid framework for dynamic balancing of exploration-exploitation: Combining regression analysis and metaheuristics. *Mathematics* **2021**, *9*, 1976.
5. Aleti, A.; Moser, I. A systematic literature review of adaptive parameter control methods for evolutionary algorithms. *ACM Comput. Surv. (CSUR)* **2016**, *49*, 1–35.
6. Zhang, J.; Sanderson, A.C. JADE: Adaptive differential evolution with optional external archive. *IEEE Trans. Evol. Comput.* **2009**, *13*, 945–958.
7. Zhan, Z.H.; Zhang, J.; Li, Y.; Chung, H.S.H. Adaptive particle swarm optimization. *IEEE Trans. Syst. Man, Cybern. Part B (Cybernetics)* **2009**, *39*, 1362–1381.
8. Liu, Y.; Zheng, L.; Cai, B. Adaptive Differential Evolution with the Stagnation Termination Mechanism. *Mathematics* **2024**, *12*, 3168.
9. Tatsis, V.A.; Parsopoulos, K.E. Reinforcement learning for enhanced online gradient-based parameter adaptation in metaheuristics. *Swarm Evol. Comput.* **2023**, *83*, 101371.
10. Eiben, Á.E.; Hinterding, R.; Michalewicz, Z. Parameter control in evolutionary algorithms. *IEEE Trans. Evol. Comput.* **1999**, *3*, 124–141.
11. Talbi, E.G. *From Design to Implementation*; John Wiley: Hoboken, NJ, USA, 2009.
12. Talbi, E.G. Machine learning into metaheuristics: A survey and taxonomy. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 129.
13. Huang, C.; Li, Y.; Yao, X. A survey of automatic parameter tuning methods for metaheuristics. *IEEE Trans. Evol. Comput.* **2019**, *24*, 201–216.
14. de Lacerda, M.G.P.; de Araujo Pessoa, L.F.; de Lima Neto, F.B.; Ludermir, T.B.; Kuchen, H. A systematic literature review on general parameter control for evolutionary and swarm-based algorithms. *Swarm Evol. Comput.* **2021**, *60*, 100777.

15. Karafotias, G.; Hoogendoorn, M.; Eiben, Á.E. Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Trans. Evol. Comput.* **2014**, *19*, 167–187.
16. Tanabe, R.; Fukunaga, A. Success-history based parameter adaptation for differential evolution. In *Proceedings of the 2013 IEEE Congress on Evolutionary Computation*; IEEE: Piscataway, NJ, USA, 2013; pp. 71–78.
17. Yang, H.; Tao, S.; Zhang, Z.; Cai, Z.; Gao, S. Spatial information sampling: Another feedback mechanism of realising adaptive parameter control in meta-heuristic algorithms. *Int. J. Bio-Inspired Comput.* **2022**, *19*, 48–58.
18. Olivares, R.; Soto, R.; Crawford, B.; Ríos, V.; Olivares, P.; Ravelo, C.; Medina, S.; Nauduan, D. A learning—based particle swarm optimizer for solving mathematical combinatorial problems. *Axioms* **2023**, *12*, 643.
19. Song, Z.; Ren, C.; Meng, Z. Differential evolution with perturbation mechanism and covariance matrix based stagnation indicator for numerical optimization. *Swarm Evol. Comput.* **2024**, *84*, 101447.
20. Prado-Rodríguez, R.; González, P.; Banga, J.R. A Parameter Control Strategy for Parallel Island-Based Metaheuristics. *Expert Syst.* **2025**, *42*, e70061.
21. Chu, P.C.; Beasley, J.E. A Genetic Algorithm for the Multidimensional Knapsack Problem. *J. Heuristics* **1998**, *4*, 63–86. <https://doi.org/10.1023/A:1009642405419>.
22. Kennedy, J.; Eberhart, R.C. A Discrete Binary Version of the Particle Swarm Algorithm. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*; IEEE: Piscataway, NJ, USA, 1997; pp. 4104–4108. <https://doi.org/10.1109/ICSMC.1997.637339>.
23. Holland, J.H. *Adaptation in Natural and Artificial Systems*; University of Michigan Press: Ann Arbor, MI, USA, 1975.
24. Goldberg, D.E. *Genetic Algorithms in Search, Optimization and Machine Learning*; Addison-Wesley: Boston, MA, USA, 1989.
25. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. <https://doi.org/10.1016/j.advengsoft.2013.12.007>.
26. Emary, E.; Zawbaa, H.M.; Hassanien, A.E. Binary Grey Wolf Optimization Approaches for Feature Selection. *Neurocomputing* **2016**, *172*, 371–381. <https://doi.org/10.1016/j.neucom.2015.06.083>.
27. Storn, R.; Price, K. Differential Evolution—A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. <https://doi.org/10.1023/A:1008202821328>.
28. Pampara, G.; Engelbrecht, A.P.; Franken, N. Binary Differential Evolution. In *Proceedings of the IEEE Congress on Evolutionary Computation*; IEEE: Piscataway, NJ, USA, 2006; pp. 1873–1879. <https://doi.org/10.1109/CEC.2006.1688535>.
29. Sakoe, H.; Chiba, S. Dynamic Programming Algorithm Optimization for Spoken Word Recognition. *IEEE Trans. Acoust. Speech Signal Process.* **1978**, *26*, 43–49. <https://doi.org/10.1109/TASSP.1978.1163055>.
30. Keogh, E.J.; Pazzani, M.J. Derivative Dynamic Time Warping. In *Proceedings of the 2001 SIAM International Conference on Data Mining*; SIAM: Philadelphia, PA, USA, 2001; pp. 1–11. <https://doi.org/10.1137/1.9781611972719.1>.
31. Clerc, M.; Kennedy, J. The Particle Swarm—Explosion, Stability, and Convergence in a Multidimensional Complex Space. *IEEE Trans. Evol. Comput.* **2002**, *6*, 58–73. <https://doi.org/10.1109/4235.985692>.
32. Vega, E.; Lemus-Romani, J.; Soto, R.; Crawford, B.; Löffler, C.; Peña, J.; Talbi, E.G. Autonomous Parameter Balance in Population-Based Approaches: A Self-Adaptive Learning-Based Strategy. *Biomimetics* **2024**, *9*, 82. <https://doi.org/10.3390/biomimetics9020082>.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.