JADIMI:
JOURNAL OF AI-DRIVEN INFORMATICS AND MANAGEMENT INFORMATION
Vol. 1 No. 1, August 2026, pp 1-11

A Hybrid Grey Wolf Optimizer-Tabu Search Approach for Multi-Day Tourist Itinerary Recommendation Based on Content-Based Filtering

Benedict Brian Joel Purba[1], Jian Hazel Sitorus[1], Dimas Putra Nugroho[2], Saskia Putri Ananda[1*]

[1] *School of Computing, Telkom University, Bandung, 40257, Indonesia*

[2] *School of Industrial Engineering, Telkom University Purwokerto, Purwokerto, 53147, Indonesia*

**Corresponding author: saskiaputria@telkomuniversity.ac.id*

Abstract*—Planning a multi-day city trip is a complex combinatorial problem that requires selecting attractions, assigning them to specific days, and determining visit order while satisfying opening hours, travel times, daily time limits, and tourist preferences and budget constraints. This study aims to develop an end-to-end recommendation system for Jakarta that integrates attraction recommendation with route optimization. The system uses open-source data from OpenStreetMap, the Massive-STEPS check-in corpus, Google Places, Wikipedia, and OSRM, producing a dataset of 166 operational attractions and 181 hotels across eight spatial clusters. In the recommendation stage, candidate attractions are ranked using TF-IDF similarity, Bayesian weighted ratings, category-based budget filtering, and Maximal Marginal Relevance to improve diversity. In the optimization stage, the Tourist Trip Design Problem is solved using a hybrid Grey Wolf Optimizer with Tabu Search (GWO–TS), adapted to permutation-based routing through swap sequences and a time-budget decoder that enforces closing hours and inserts a lunch break. GWO–TS is compared with a genetic algorithm, particle swarm optimization, and four additional metaheuristic hybrids using fitness, user satisfaction, constraint violations, and runtime over fixed seed-paired independent runs. Across 630 solver runs, all seven methods show statistically similar user satisfaction. GWO–TS is statistically comparable with the strongest methods in penalty-adjusted fitness while being significantly faster than every competitor. These results indicate that GWO–TS provides a favorable balance between solution quality and computational efficiency, supporting its deployment in interactive trip-planning services. Future research should examine broader destinations, richer contextual preferences, and more adaptive real-time optimization.

Keywords—TTDP; itinerary recommendation; GWO; TS; CBF; metaheuristics; spatial clustering

Manuscript received 11 Aug. 2026; revised 20 Aug. 2026; accepted 23 Aug. 2026. Date of publication 30 Aug. 2026.

I. INTRODUCTION

Independent city tourism increasingly relies on digital assistants to turn a vague intent ("two relaxed days of museums, mid-range budget") into a concrete, feasible plan. For a large and congested metropolis such as Jakarta, this is not merely ranking attractions: it requires deciding which places to include, in what order, and on which day, subject to per-venue opening hours, realistic travel time, a bounded daily schedule, and the visitor's preference and budget. This is a personalized Tourist Trip Design Problem (TTDP), an orienteering-type route problem that is NP-hard [1], [2].

Despite a large literature on tourism recommendation and route optimization, this study observes three gaps that motivate this work. (G1) Multi-day planning. Many systems recommend only a single day or a ranked POI list, whereas real trips span several days with cross-day spatial and temporal coupling [3], [4]. (G2) User satisfaction. Route solvers often minimize travel while under-modeling how satisfying the resulting plan is to the user; satisfaction deserves to be an explicit objective and evaluation target [5]. (G3) Implementation machine learning with optimization. Preference learning (ML) and route optimization

are frequently studied in isolation rather than integrated in one working pipeline [6], [7].

This paper addresses these gaps with an end-to-end system for Jakarta that couples a content-based filtering (CBF) stage with a metaheuristic route optimizer, and introduces GWO–TS, a hybrid Grey Wolf Optimizer with an embedded Tabu Search local search, as the optimization core. This paper's contributions are:

- A multi-day TTDP formulation (G1) with a time-budget decoder that enforces opening hours as a hard constraint and penalizes spatial incoherence.
- A satisfaction-centric objective (G2): a content model produces a per-attraction satisfaction score that the optimizer maximizes, and a normalized User Satisfaction Score is used for evaluation.
- An integrated ML-plus-optimization pipeline (G3): TF-IDF similarity, a Bayesian rating, and spatial clustering feed a discrete GWO–TS optimizer, all built on open data and deployed behind a service.
- An empirical comparison of GWO–TS against a genetic algorithm (GA) baseline and five further metaheuristics on four metrics such as fitness value, running time, convergence and user satisfaction score.

Section II reviews related work, Section III describes the pipeline and the GWO–TS optimizer, Section IV reports and discusses results, and Section V concludes.

II. RELATED WORK

The TTDP generalizes the orienteering and team-orienteering problems, where a subset of prized nodes is visited within a budget to maximize collected prize [1]. Recent formulations add time windows, multiple tours (days), and personalization, and solve them with heuristics and metaheuristics [2], [3]. However, a substantial share of deployed tourism recommenders still targets single-day or list-style output [4], [5], leaving multi-day, cross-day coupling underserved. The authors address this with K-Means and a soft inter-zone penalty.

POI recommenders span collaborative, content-based, context-aware, and tensor-factorization models [7]. Content-based filtering scores items by profile similarity and is attractive under sparse interaction data and available item text [7]. Diversity-aware re-ranking trades relevance against novelty to avoid near-duplicate suggestions [8]. Yet many route-centric systems optimize distance or time while treating user satisfaction only implicitly [5]; this work instead makes satisfaction the reward signal and an explicit metric, and standard recommender metrics inform the content stage [9].

Population metaheuristics GA [10], PSO [11], ACO [12], WOA [13] and trajectory methods such as simulated annealing [14] and Tabu Search are staples for combinatorial routing, with active recent development and hybridization [15],[16],[17]. The Grey Wolf Optimizer guides a population with three leaders (alpha, beta, delta) and a control parameter that shifts search from exploration to exploitation using few parameters [18]; improved and discrete variants report strong performance on engineering and routing problems [19], [20]. Because population methods explore globally but refine local order weakly, they are combined with local search into memetic hybrids [21], [22]. Separately, integrating ML-based preference learning with metaheuristic optimization in a single pipeline remains comparatively rare [6], [23]. This GWO–TS follows the hybrid tradition and closes the ML optimization loop: a content model supplies the objective and GWO–TS, with Tabu Search intensification, optimizes the multi-day route.

III. METHODOLOGY

The system has two phases behind a deployed service: Phase 1 (content-based filtering) scores and diversifies candidate attractions; Phase 2 (route optimization) arranges them into a multi-day itinerary with GWO–TS. The same modeling code drives both an offline experiment harness and the online API, so the evaluated and served models coincide.

A. Data and Pipeline

The dataset is built entirely from open sources OpenStreetMap (coordinates, categories, the DKI Jakarta boundary), the Massive-STEPS Jakarta check-in corpus (412,100 check-ins for popularity), Google Places (opening hours, ratings, descriptions), Wikipedia (description enrichment), and OSRM (road travel times) through a cache-aware ETL that deduplicates, filters to a tourism whitelist, merges by proximity, clips to the province polygon, and consolidates the large TMII complex into one destination. The result is 166 operational attractions and 181 hotels across 8 K-Means spatial zones, with 13,695 all-pairs OSRM travel times (plus a non-toll matrix for the motorcycle mode).

The integration proceeds in three stages, each implemented as an idempotent, cache-aware step so that the dataset can be rebuilt from the published scripts. Collection harvests candidate venues from the OpenStreetMap Overpass API, restricted to the DKI Jakarta administrative boundary at admin_level 4 rather than to a rectangular bounding box, and filtered to a tourism whitelist defined over the tourism, leisure, historic and amenity keys. In parallel, the Massive-STEPS Jakarta corpus contributes 412,100 real check-ins as the popularity signal, and the Google Places API is queried across twelve tourism categories and twenty-one district anchors for opening hours, ratings and descriptions, together with a pool of 280 candidate hotels.

Cleaning then removes exact and near-duplicate records, restricts the Massive-STEPS venues to a fourteen-category tourism whitelist over the Foursquare taxonomy, and applies keyword and explicit-name

exclusion lists compiled from manual spot checks. This second curation layer is necessary because Foursquare categories are user-assigned rather than ground truth: records that pass category filtering but are not tourist destinations, such as aquarium retailers, arcade outlets inside shopping malls, public swimming pools, private clubhouses, travel agencies and venues whose coordinates proved inconsistent with their names, are removed at this point.

Merging joins the Massive-STEPS venues to OpenStreetMap records within a 150 m radius, then enriches the merged table with Google Places opening hours, ratings, addresses and descriptions, backfilling any remaining empty description from Wikipedia so that the TF-IDF representation in (1) is defined for every attraction. A final pass clips the result to the DKI Jakarta polygon, retains only venues whose Google business status is operational, and consolidates the forty sub-venues inside the Taman Mini Indonesia Indah complex into a single parent destination through a geofence, since they share one admission ticket and are experienced as one visit. Each surviving attraction is assigned an estimated on-site duration, which the decoder of Section III.C consumes as its visit time.

B. Phase 1: Content-Based Filtering

Each attraction is represented by its category and description via TF-IDF; the relevance of attraction i to a preference query q is the cosine similarity, as shown in (1).

$$sim(q,i) = (q \cdot d_i) / (\|q\| \, \|d_i\|) \quad (1)$$

Because raw ratings favor obscure venues, popularity uses a Bayesian (IMDB-style) weighted rating, as shown in (2).

$$WR_i = [v_i / (v_i + m)] \cdot R_i + [m / (v_i + m)] \cdot C \quad (2)$$

With rating R_i, rating count v_i, median count m, and global mean rating C; WR_i is normalized to $p_i \in [0, 1]$. A category price proxy filters by budget. The satisfaction score fed to the optimizer, as shown in (3).

$$s_i = w_{sim} \cdot sim(q,i) + w_{po} \quad (3)$$
$$= 0.6, w\_pop = 0.4$$

Candidates are selected by Maximal Marginal Relevance[24], $MMR(i) = \lambda s_i - (1 - \lambda) \cdot max_j \in s \; sim(i,j) \; with \; \lambda = 0.7$, which trades relevance against redundancy to keep the pool diverse in the sense of [8].

The three weights of the content stage were fixed a priori rather than swept, and the reasoning is reported here for transparency. In (3), w_sim = 0.6 and w_pop = 0.4 place the majority of the weight on the stated preference so that the query, not the crowd, drives the ranking, while retaining a substantial popularity term because similarity alone promotes thinly documented venues whose short description happens to echo the query wording. The Bayesian correction in (2) already suppresses the opposite failure mode, in which a venue holding a small number of very high ratings outranks a major landmark rated by tens of thousands of visitors. For the diversification step, lambda = 0.7 was chosen because the corpus contains clusters of near-identical descriptions: before consolidation, the Taman Mini complex alone contributed twenty-one pavilions whose text is almost interchangeable, and a purely relevance-ordered pool, obtained at lambda = 1, filled with such duplicates and forced the optimizer to split a single complex across several days. A value of 0.7 keeps relevance dominant while penalising redundancy enough to break those clusters. Because these three values were not tuned, the sensitivity of the satisfaction ranking to them is recorded as a limitation in Section IV.H.

C. Phase 2: Multi-Day TTDP

A solution is a permutation of candidate indices. A time-budget decoder walks the permutation from the hotel at day start, adding travel and visit time, inserting a mandatory lunch break, and returning before day end.

An attraction that would arrive after closing is deferred to a later day or left unvisited. Concretely, each day runs from 08:00 to 20:00. The lunch break is inserted within an 11:30–13:30 window: once the running schedule reaches this window without a break already placed, a fixed 60-minute break is inserted before the next visit, reducing the effective daily time budget by the same amount. The candidate pool passed to Phase 2 scales with trip length at twelve candidates per day, so a three-day request draws from up to 36 CBF-ranked candidates; the system caps requests at five days, the upper bound used for the multi-day scenarios in Section IV.

Closing hours are thus a hard constraint. Let the decoder yield the visited set V(x), travel time T(x) (min), inter-zone transitions Z(x), intra-day zone revisits $Z_r(x)$, cross-day revisits $Z_{d(x)}$, and hour violations H(x). Fitness (maximized), as shown in (4).

$$f(x) = \Sigma_i \in V(x) s_i - w_T \cdot \frac{T(x)}{60} - w_Z \cdot Z(x) - w_R \cdot Z_r(x) - w_D \cdot Z_{d(x)} - w_H \cdot H(x) \quad (4)$$

With $w_T = w_Z = w_D = 1, w_R = 3$ (heavier weight discourages intra-day back-and-forth), and $w_H = 5$. All optimizers share this decoder, fitness, and a 2-opt/TS polish, so comparisons isolate the search strategy.

To make the two zone-coherence penalties concrete, $Z_r(x)$ and $Z_{d(x)}$ track different failure modes over the same permutation. $Z_r(x)$ increments whenever the itinerary returns to a zone already left earlier the same

day. For instance, a schedule that moves Zone 0 → Zone 5 → Zone 0 counts one intra-day revisit while $Z_{d(x)}$ increments when a zone visited on an earlier day is revisited on a later one, capturing fragmentation across the whole trip rather than within a single day.

A candidate that cannot be reached before closing under the current permutation is not discarded outright: consistent with Section III.E, the decoder first tries to place it on a later day with remaining capacity before marking it unvisited, so the itinerary returned to the client distinguishes a scheduled visit from a candidate that simply did not fit.

D. GWO-TS Optimizer

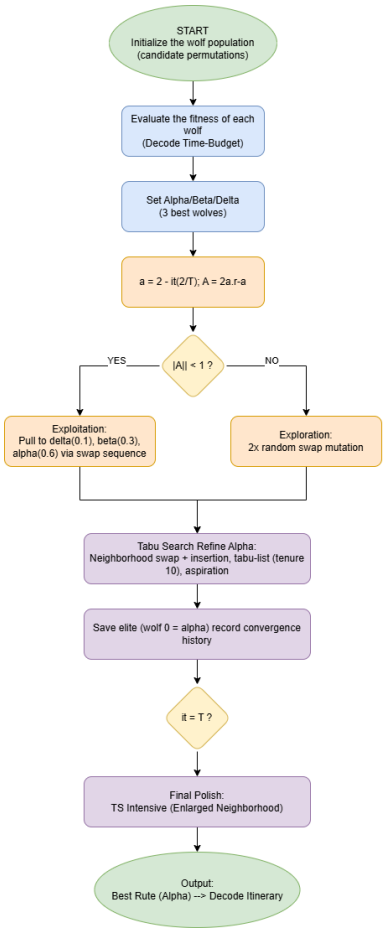

Figure. 1 GWO–TS control flow: leader-guided global search with an exploration/exploitation switch, per-iteration Tabu Search on the alpha wolf, and a final polish

Continuous GWO moves each wolf toward the alpha/beta/delta leaders with a coefficient a decreasing from 2 to 0 [18]. This paper adapts it to the permutation space using swap sequences (the transpositions that map one permutation to another): applying a fraction of a leader's swap sequence pulls a wolf part-way toward it. At iteration t of T, as shown in (5).

$$a = 2 - t(2/T),\ A = 2ar - a,\ r \sim U(0,1) \qquad (5)$$

If |A| < 1 (exploitation), a wolf is pulled toward delta, beta, and alpha with increasing acceptance p_δ = 0.1, p_β = 0.3, p_α = 0.6 (the best leader pulls hardest); if |A| ≥ 1 (exploration), it receives two random swap mutations. Wolf 0 is an elite: never moved, overwritten by the Tabu-Search-refined alpha each iteration, so incumbent fitness is monotone.

Population methods refine local order weakly, so the alpha is intensified every iteration with Tabu Search (TS): from a neighborhood mixing a random swap and a random insertion, the best non-tabu neighbor is accepted (a tabu one only if it beats the local best aspiration); a tabu list of fixed tenure forbids cycling. A stronger final TS polish closes the run. TS thus plays the role a 2-opt polish plays for the baselines but is applied every iteration to the leader. Algorithm 1 and Fig. 1 summarize the method.

Hybrid combinations of Grey Wolf Optimization and Tabu Search have been reported before, most directly for cold-chain logistics distribution [22]. The contribution claimed here is therefore not the pairing itself but the way both components are redefined for the multi-day TTDP, and it rests on four specific differences. First, the wolf update is not a continuous position update but a swap-sequence operator over permutations: the transposition sequence mapping a wolf to a leader is generated and then applied element-wise with acceptance probabilities that rise from delta to beta to alpha, so a wolf is drawn part-way toward each leader in proportion to that leader's rank rather than displaced by a scalar coefficient. Second, Tabu Search is not a terminal refinement stage but a per-iteration intensification applied exclusively to the incumbent alpha, over a neighbourhood mixing swap and insertion moves; the insertion move is retained because relocating a single attraction changes the day on which it falls under the time-budget decoder, a transformation that a pure swap neighbourhood cannot express and that is specific to the multi-day setting. Third, the tabu list stores whole permutation tuples under an aspiration criterion, and the refined alpha is written back into a reserved elite wolf that the population update never displaces, which makes incumbent fitness monotone across iterations. Fourth, the objective being optimised is not a distance or a monetary cost but the satisfaction score produced by the content stage in (3), so the search is coupled to a learned preference model rather than to a purely geometric criterion.

Algorithm 1 GWO–TS for the multi-day TTDP

Require: wolves M, iterations T, pulls pδ, pβ, pα, tenure τ
1: init population; evaluate f; set α, β, δ; L ← ∅
2: for t = 0 to T − 1 do
3: a ← 2 − t(2/T)
4: for each non-elite wolf x do
5: A ← 2a r − a
6: if |A| < 1 then
7: pull x toward δ, β, α using pδ, pβ, pα
8: else
9: apply two random swap mutations to x

10: end if
11: end for
12: evaluate f; update α (best-so-far), β, δ
13: α ← TabuRefine(α, L, τ); elite wolf ← α
14: record f(α) ▷ convergence history
15: end for
16: final TS polish of α; return α and its decoded itinerary

E. System Architecture and Deployment

The pipeline is exposed as a two-tier deployed service. The backend is a FastAPI application (src/api/api.py) that reuses the same modeling code used for the offline experiment harness, so the models evaluated in Section IV are exactly the models served in production. It exposes three endpoints: GET /venues, which returns the attraction catalog with category, zone, rating, price level, and coordinates; GET /hotels, which returns the candidate departure/return points; and POST /itinerary, which accepts a preference query, budget tier, number of days, start day, and hotel choice, and returns a day-by-day schedule with per-visit timestamps, travel segments, lunch breaks, aggregate fitness, and any candidates that could not be fitted into the itinerary. Google Places' price_level field is missing for nearly all venues in the dataset, the budget filter instead uses a category-based price proxy with four discrete levels: 0 (free), 1 (inexpensive, under Rp 50,000), 2 (moderate, Rp 50,000–150,000), and 3 (expensive, over Rp 150,000). A tourist's budget tier such as hemat (economical), menengah (mid-range), or bebas (flexible), maps to an inclusive price-level ceiling of 1, 2, and 3 respectively, so a mid-range request admits any candidate at level 2 or below.

The service supports two request modes. In automatic mode, the client supplies only a free-text preference (e.g., "museum sejarah budaya") and a budget tier; Phase 1 (Section III.B) selects and scores the candidate pool via TF-IDF, the Bayesian weighted rating, and MMR diversification. In manual mode, the tourist selects venues directly from the catalog and the service is responsible only for Phase 2 sequencing. Both modes share the same time-budget decoder and optimizer.

The optimizer is selectable at request time. The algorithm field accepts ga, pso, hybrid (GA–PSO) or gwo_ts explicitly, which is what supported the controlled comparison in Section IV, and defaults to auto. In production auto resolves to GWO–TS for every request, irrespective of trip length, replacing the GA–PSO hybrid that the service previously deployed. That default follows from the experimental results rather than from a preference: GWO–TS is statistically indistinguishable from the strongest competitors on both penalty-adjusted fitness and user satisfaction (Section IV.G), holds the highest mean USS aggregated over the scenario grid, and is significantly faster than every alternative, which is the property that governs whether the service can answer within an interactive session. The explicit algorithm values are retained so that the comparison reported here can be reproduced against the deployed system. A React (Vite) frontend, branded Jakarta Routes, consumes this API and overlays the resulting itinerary on an interactive Leaflet map using real OSRM road geometry, with distinct colors per day and dashed segments for the final approach to each venue.

IV. EXPERIMENTAL RESULT AND DISCUSSION

A. Setup

The authors evaluate on a representative three-day city-break scenario, the common multi-day case scheduled over a weekend (Friday to Sunday) across three distinct spatial zones (Zone 0, Zone 5, and Zone 2), with the user preference set to theme parks, zoos, and aquariums. GWO–TS is compared against a genetic algorithm (GA) as the primary baseline and five further metaheuristics, namely Hybrid ACO–SA, Hybrid GA–PSO, Hybrid GRASP–ABC, PSO, and Hybrid WOA–ILS. All solvers optimize the same TTDP instance, with an identical candidate set, satisfaction scores, decoder and fitness function (4), so differences reflect only the search strategy. The experimental campaign covers all seven optimizers across all three scenarios at 30 runs each, giving 630 solver executions and 90 paired blocks. The study reports four metrics motivated by the research gaps: fitness value (solution quality and satisfaction), User Satisfaction Score (how much predicted preference survives the constraints), running time (interactive feasibility), and convergence behaviour.

Seed handling deserves an explicit statement, because the manuscript refers both to multiple independent runs and to a fixed random seed, and these describe two different levels of the same protocol rather than a contradiction. A single base seed is declared as a constant in the shared configuration file. The seed actually passed to a solver on run r is that base seed offset by the run index, so a campaign of R runs draws on the fixed, contiguous seed set {base, base+1, …, base+R-1}. The campaign as a whole is therefore reproducible from one constant, while the runs within it remain independent of one another. The same seed set is supplied to every algorithm, so observations are paired by seed: run r of GWO–TS and run r of the GA baseline begin from populations drawn from the same random stream. That pairing is what licenses the blocked statistical treatment described in Section IV.G.

Beyond the three-day case reported in detail, the experiment harness is defined over a scenario grid that varies trip length, preference text and budget ceiling together: a one-day museum-and-heritage trip at the mid-range budget tier, the three-day family scenario described above at the unrestricted tier, and a five-day mixed scenario combining museums, parks, beaches and monuments at the mid-range tier. Because the candidate pool scales at twelve candidates per day, these three cases also present search spaces of markedly different size. The

three-day case is reported in full here because it is the modal city-break length and exercises cross-day spatial coupling without saturating the daily time budget. All three are reported: Table I gives the three-day case in detail, and the statistical tests of Section IV.G are computed over blocks drawn from all three, so the significance results are not conditional on a single trip length. Further variation in spatial zoning and budget ceiling remains part of the extended evaluation identified in Section V.

Table I summarizes the three-day scenario and Table II lists the deployed settings; Tables IV to VI report satisfaction and the statistical tests. Fig. 2 shows fitness, satisfaction and runtime together, Fig. 3 the satisfaction detail, and Fig. 4 the convergence.

TABLE I
OPTIMIZER COMPARISON ON THE THREE-DAY SCENARIO OVER 30 PAIRED RUNS. ROWS ORDERED BY MEAN FITNESS

Algorithm	Fitness	Std	USS	Visited	Time (s)
Hybrid WOA–ILS	0.914	1.065	0.9448	21.7	7.28
GA (baseline)	0.288	0.599	0.9426	23.8	4.92
Hybrid GWO–TS	0.280	0.974	0.9451	23.5	3.37
Hybrid GA–PSO	0.222	0.883	0.9442	23.4	7.10
PSO	0.052	0.799	0.9428	22.8	7.51
Hybrid ACO–SA	−2.282	2.018	0.9453	20.5	5.70
Hybrid GRASP–ABC	−2.973	1.405	0.9452	19.2	27.66

TABLE II
OPTIMIZER CONFIGURATIONS (DEPLOYED SYSTEM)

Optimizer	Settings
GA (baseline)	pop 50, gen 200, pc=0.7, pm=0.3, tourn. 3, elite 2
PSO	50 particles, 200 iter, w=0.5, c1=c2=1.5
GWO–TS	50 wolves, 120 iter, pulls 0.1/0.3/0.6; TS tenure 10, 16 neighbors, 3 polish passes
All shared	decoder, fitness, polish; seed = base + run index
Hybrid GA–PSO	Reuses PSO settings (50 particles, 200 iter, w=0.5, c1=c2=1.5) with a genetic refresh applied every 10 iterations

All optimizers share the same permutation-based encoding, time-budget decoder and fitness function (4), so the configuration in Table II isolates each algorithm's own search behaviour rather than differences in problem formulation. The values in Table II were not adopted unchanged from the literature. They were selected by a grid search run under a common protocol: the computational budget was held constant across every combination (population or swarm size 50, and 200 generations or iterations), the fitness function was excluded from the search because altering it would alter the metric itself, and the shared final polish was disabled during tuning so that the measurement reflects the core search operators rather than a common post-processing stage. Each combination was evaluated over five runs on two deliberately dissimilar scenarios, the smallest and the largest in the scenario grid, and the winning combination was chosen by mean rank across those two scenarios rather than by mean fitness, since the fitness scales of a one-day and a five-day instance are not directly comparable.

The grids were as follows: mutation rate crossed with crossover rate over {0.1, 0.2, 0.3} × {0.7, 0.8, 0.9} for GA; inertia crossed with the acceleration coefficients over {0.4, 0.5, 0.7} × {1.0, 1.5, 2.0} for PSO; the genetic refresh interval over {5, 10, 20} for the GA–PSO hybrid, evaluated at the best PSO coefficients; and, for GWO–TS, the alpha pull probability crossed with the Tabu neighbourhood size over {0.6, 0.8} × {8, 16, 24}, with the beta and delta pull probabilities scaled from the alpha value so that the progressive ordering p_delta < p_beta < p_alpha is preserved. The most informative single outcome concerned GWO–TS, where enlarging the Tabu neighbourhood from 8 to 16 improved fitness substantially on the largest scenario, confirming that the intensification stage rather than the leader-guided global move is what repairs travel-inefficient orderings. GWO–TS retains 120 rather than 200 iterations because that same per-iteration intensification converges the leader earlier, an effect visible in Fig. 4.

One asymmetry must be stated plainly. This tuning protocol was applied to the four optimizers implemented in the study's own codebase, namely GA, PSO, the GA–PSO hybrid and GWO–TS. The three remaining comparators were evaluated at their reported configurations and were not re-tuned under an identical grid. The comparison is therefore symmetric in problem formulation, computational budget and seed set, but not in tuning effort, and the consequences of that are discussed in Section IV.H.

B. Fitness Value

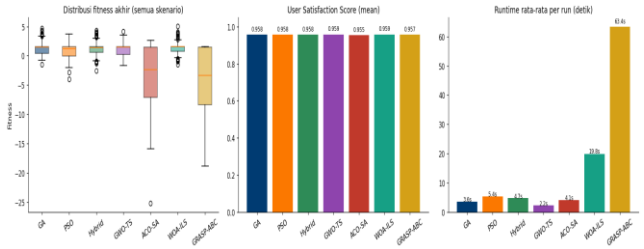

Figure. 2 Fitness distribution, mean User Satisfaction Score, and mean runtime per optimizer over all 630 runs. GWO–TS holds a top-tier fitness median with a tight spread and the lowest runtime.

Table I summarizes the three-day scenario over 30 paired runs. Fitness on this instance is a net score, satisfaction collected minus travel and zoning penalties, so it can fall below zero when an optimizer accumulates penalties faster than satisfaction. Five of the seven methods stay positive and cluster tightly: WOA–ILS at

0.914, GA at 0.288, GWO–TS at 0.280, GA–PSO at 0.222 and PSO at 0.052. ACO–SA and GRASP–ABC fall well below zero, at –2.282 and –2.973, because both construct routes greedily by proximity and satisfaction without a mechanism that penalises returning to a zone already left, so they accumulate the inter-zone and revisit penalties of (4). GWO–TS visits 23.5 attractions on average, close to the highest count in the table, and does so in the shortest time. The behaviour follows from its design: the alpha/beta/delta hierarchy provides directional guidance with few parameters, while the per-iteration Tabu Search repairs the fine ordering of the leader and the elite wolf enforces monotone improvement.

The correct characterisation of this result is a balance rather than a victory, and the statistical tests of Section IV.G make the point precisely. GWO–TS is not the highest-scoring method on fitness: its mean rank across all 90 blocks places it fifth of seven. What the post-hoc tests establish is that it is not statistically distinguishable from any of the four methods above it, while being distinguishable from the two below. GWO–TS therefore belongs to the leading group on solution quality rather than leading it, and its distinctive property emerges only once computational cost is placed alongside quality, which is the subject of Section IV.D. Throughout the remainder of this section GWO–TS is described as offering a favourable balance between competitive solution quality and computational efficiency, not as the best-performing method on quality.

C. User Satisfaction and Constraint Violations

The fitness in (4) is not an abstract objective. Its only reward is the sum of per-attraction satisfaction scores $\Sigma_{i \in V(x)}$ from the content model (3); every other term is a penalty for travel or for a constraint breach. A higher fitness therefore directly denotes a more satisfying and more constraint-adherent itinerary, so the fitness ranking of Table I is equivalently a ranking by user satisfaction under constraints. To make this explicit and scale-free, this study reports the User Satisfaction Score (USS), the collected satisfaction relative to the ideal top-K for the same number of visited attractions, as shown in (6).

$$USS(x) = [\Sigma_i \in V(x)\ s_i] \,/\, [\Sigma_i \in top - |V(x)|\ s_i] \in [0, 1] \quad (6)$$

This score increases monotonically with the satisfaction component of (4) and is bounded in [0, 1], so unlike raw fitness it is comparable across scenarios of different length. It is computed and logged for every run of every algorithm from the visited set returned by the decoder, so a per-algorithm mean and standard deviation is available on exactly the same footing as the fitness columns. Table III reports these figures over all 630 runs. GWO–TS attains the highest mean, 0.9586, and simultaneously the smallest standard deviation, 0.0207, so it is both the most satisfying and the most consistent of the seven on this metric.

That advantage should not be overstated. The spread between the highest and lowest mean in Table III is 0.0037, roughly one sixth of a single standard deviation, and the Friedman test on USS reported in Section IV.G does not reject the null hypothesis of equivalence. The honest reading is that all seven optimizers preserve essentially the same fraction of predicted satisfaction, and that the differences between them lie elsewhere. This is itself informative: the spatial and temporal constraints, not the choice of search strategy, are what determine how much predicted preference survives into a feasible itinerary. On average every method retains above 95 per cent of the unconstrained ideal, so the constraint structure of Section III.C is not the bottleneck a tourist typically experiences, though individual runs fall as low as 0.893.

TABLE III
USER SATISFACTION SCORE PER OPTIMIZER (630 RUNS)

Algorithm	Mean	Std	Min	Max	Rank
Hybrid GWO–TS	0.9586	0.0207	0.919	0.995	3.88
Hybrid WOA–ILS	0.9585	0.0273	0.897	0.995	3.80
Hybrid GA–PSO	0.9581	0.0229	0.916	0.995	3.92
PSO	0.9578	0.0251	0.913	0.995	3.94
GA (baseline)	0.9575	0.0220	0.921	0.995	4.08
Hybrid GRASP–ABC	0.9569	0.0220	0.912	0.995	4.14
Hybrid ACO–SA	0.9549	0.0282	0.893	0.995	4.24

One clarification matters for the interpretation of this metric. USS as defined in (6) is a model-internal quantity. It measures how much of the satisfaction that the content stage predicts for a given tourist is actually retained once the spatial and temporal constraints are enforced, and it is therefore a measure of the optimizer's ability to preserve predicted preference relative to an unconstrained ideal, not a measurement of experienced satisfaction. The content stage itself was validated offline against observed behaviour, by leave-one-out evaluation over the Massive-STEPS check-in histories of users with at least two visited venues, scored with HitRate@K, nDCG@K and MRR against popularity-only and random baselines; but that exercise validates the ranking model, not the generated itineraries. A small-scale study with real tourists, comparing generated itineraries against self-reported satisfaction, would supply the complementary perspective and is identified as future work in Section V.

Regarding constraint violations, the opening-hour term H(x) (4) is zero for every solver by construction: the

time-budget decoder enforces closing hours as a hard constraint, deferring or dropping any visit that cannot be served in time, so all methods return time-valid itineraries.

Feasibility is thus guaranteed rather than optimized and is not a differentiator; what the remaining penalties (w_Z, w_R, w_D) do differentiate is spatial coherence, inter-zone hops and same-day zone revisits, so a higher-fitness plan is not only feasible but also more compact and less zig-zagging, the qualities a traveler actually experiences.

D. Running Time

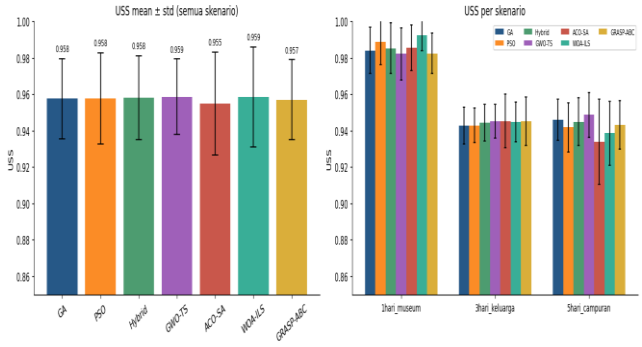

Figure. 3 User Satisfaction Score per optimizer, aggregated and per scenario (error bars: one standard deviation). The differences are not statistically significant (Table V).

Runtime is where the seven methods separate unambiguously. On the three-day scenario GWO–TS completes a run in 3.37 s against 4.92 s for the GA baseline, 5.70 s for ACO–SA, 7.10 s for GA–PSO, 7.28 s for WOA–ILS, 7.51 s for PSO and 27.66 s for GRASP–ABC. Pooled across all three scenarios the mean rank of GWO–TS is 1.69, the best of the seven, and the Wilcoxon post-hoc results in Table VI show that it is significantly faster than every one of the six competitors at the Bonferroni-corrected level. This is the one axis on which the method dominates outright, and it is the axis that governs whether a recommender can answer interactively. Two mechanisms explain it: GWO uses few control coefficients per iteration, and the Tabu intensification converges the leader early enough that 120 iterations suffice where GA and PSO are given 200.

E. Convergence

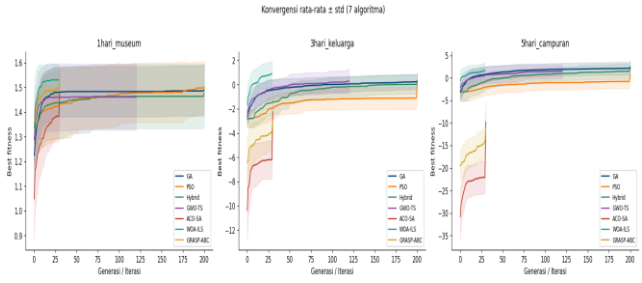

Figure. 4 Mean convergence with one-standard-deviation bands for the seven optimizers on each scenario. GWO–TS front-loads improvement and stabilizes well before its 120-iteration budget.

Fig. 4 contrasts convergence across all seven optimizers on each scenario. GWO–TS front-loads improvement, approaching its final fitness within roughly the first thirty iterations, whereas GA and PSO continue to climb through much of their 200-generation budget. The abrupt termination of the ACO–SA, WOA–ILS and GRASP–ABC curves at iteration 30 is their iteration budget, not convergence, and is the limitation recorded in Section IV.H. The early plateau of GWO–TS is a direct effect of intensifying a single strong leader with Tabu Search at every step rather than relying solely on population-level recombination, and it is what allows the 120-iteration budget to suffice.

F. Comparison of GA and the Other Baselines

TABLE IV
GA BASELINE VERSUS HYBRID GWO–TS (THREE-DAY SCENARIO)

Algorithm	Fitness	Std	USS	Visited	Time (s)
GA (baseline)	0.288	0.599	0.9426	23.8	4.92
Hybrid GWO–TS	0.280	0.974	0.9451	23.5	3.37

Against the GA baseline specifically, Table IV shows a split rather than a sweep. On the three-day scenario the two are level on fitness, 0.288 for GA against 0.280 for GWO–TS, a gap far smaller than either standard deviation and not significant under the Wilcoxon test ($p = 0.0124$ against a corrected threshold of 0.0024). GWO–TS retains slightly more predicted satisfaction (0.9451 against 0.9426) while visiting a comparable number of attractions, and it completes a run in about two thirds of the time. The defensible claim is equivalence on quality and a significant advantage on cost. GA converges reliably but its recombination-only search lacks the targeted local repair that TS provides, leaving a quality and speed gap. Briefly, the remaining methods trade quality for cost in different ways. Hybrid WOA–ILS attains the highest mean fitness on the three-day scenario (0.914) and the best mean rank on fitness overall, but its runtime is erratic, averaging 19.76 s across scenarios with a standard deviation of 23.00 s, which makes worst-case response time hard to bound in a service. Hybrid GA–PSO tracks the GA baseline closely on quality at roughly twice the runtime of GWO–TS. PSO converges cheaply but to the lowest mean among the five positive methods, reflecting weak local refinement. Hybrid ACO–SA and Hybrid GRASP–ABC are the two methods that fall below zero, and GRASP–ABC is also the slowest overall, averaging 63.40 s when pooled across the three scenarios against 27.66 s on the three-day case of Table I, since its cost grows steeply with trip length; both, however, run at a 30-iteration budget rather than the 120 to 200 of the others, so their position in Table I reflects that budget as much as their search behaviour. Overall, GWO–TS offers the best balance of the three target metrics.

G. Statistical Significance

Because the comparison rests on repeated independent runs rather than on a single execution, the ranking in Table I is only meaningful if the differences between algorithms survive a test of statistical significance. The design adopted here is the paired non-parametric one that the protocol of Section IV.A admits. Fitness values across runs are not assumed to be normally distributed, and the seed handling described above means the observations are naturally blocked: each combination of scenario and run index forms a block within which every algorithm was evaluated on the same random stream. The Friedman test is therefore applied as the omnibus test over these blocks, under the null hypothesis that all algorithms are equivalent in fitness.

Where the omnibus test rejects that null at the 0.05 level, pairwise Wilcoxon signed-rank tests are applied post hoc, with a Bonferroni correction over the number of pairs to control the family-wise error rate. Where it does not reject, no post-hoc test is performed, since pairwise comparisons conditioned on a non-significant omnibus result are not interpretable. Mean ranks per algorithm are reported alongside the test statistic in either case, because the rank profile remains informative even when the omnibus result is negative. A non-significant outcome would not invalidate the contribution of this work: it would indicate that the compared methods are statistically indistinguishable on solution quality for this instance, which strengthens rather than weakens the argument of Section IV.D, since the selection of an optimizer would then rest legitimately on the secondary criteria of runtime and run-to-run stability.

Table V reports the omnibus results over the 90 blocks. The three metrics behave differently, and that difference is the central empirical finding of this work. On fitness the null hypothesis is rejected decisively; on runtime it is rejected even more strongly; on USS it is not rejected at all ($p = 0.7854$). The seven optimizers are therefore statistically indistinguishable in how much predicted satisfaction they deliver, clearly distinguishable in the penalty-adjusted fitness they achieve, and clearly distinguishable in the time they take.

TABLE V

FRIEDMAN OMNIBUS TEST (BLOCKS = SCENARIO × RUN)

Metric	Blocks	Chi-square	p-value	Significant	Rank 1
Fitness	90	200.23	1.7×10^{-40}	yes	WOA–ILS (2.71)
USS	90	3.18	0.7854	no	WOA–ILS (3.80)
Runtime (s)	90	309.61	7.1×10^{-64}	yes	GWO–TS (1.69)

Because the omnibus test rejects the null on fitness and on runtime, pairwise Wilcoxon signed-rank tests were applied post hoc over the 21 pairs, with the Bonferroni-corrected threshold $0.05/21 = 0.0024$. Table VI condenses the results to the comparisons involving GWO–TS. On fitness, none of the four differences against GA, PSO, GA–PSO and WOA–ILS survives correction, so GWO–TS cannot be separated from any of them; the differences against ACO–SA and GRASP–ABC do survive, and favour GWO–TS. On runtime, every one of the six comparisons is significant and every one favours GWO–TS. No post-hoc test is reported for USS, since pairwise comparisons conditioned on a non-significant omnibus result are not interpretable.

TABLE VI

WILCOXON POST-HOC: GWO–TS VERSUS EACH COMPETITOR

Comparison	Fitness p	Sig.	Runtime p	Sig.	Faster
vs GA (baseline)	0.0124	no	$< 10^{-16}$	yes	GWO–TS
vs PSO	0.3093	no	$< 10^{-16}$	yes	GWO–TS
vs Hybrid GA–PSO	0.0960	no	$< 10^{-16}$	yes	GWO–TS
vs Hybrid WOA–ILS	0.0140	no	$< 10^{-10}$	yes	GWO–TS
vs Hybrid ACO–SA	$< 10^{-12}$	yes	$< 10^{-9}$	yes	GWO–TS
vs Hybrid GRASP–ABC	$< 10^{-12}$	yes	$< 10^{-16}$	yes	GWO–TS

Read together, the three tests give a sharper statement than any single ranking. A reader who cares only about penalty-adjusted fitness has no statistical basis for preferring any of the top five methods over another. A reader who cares about the satisfaction actually delivered to a tourist has no basis for preferring any of the seven. Once response time is admitted as a criterion, the ambiguity disappears, and it disappears in favour of GWO–TS. A negative omnibus result on USS is therefore not a weakness of this work but the condition that makes its argument valid: it is precisely because the methods are equivalent on delivered satisfaction that a decision may legitimately rest on cost.

H. Threats to Validity

Four limitations bear directly on the strength of the experimental conclusions. First, the iteration budget is not uniform across the seven methods. GA, PSO and GA–PSO run for 200 generations and GWO–TS for 120, while ACO–SA, WOA–ILS and GRASP–ABC run for 30 iterations, the budget at which they were originally specified and reproduced here for fidelity to their source implementations. The poor fitness of ACO–SA and GRASP–ABC in Table I should therefore not be read as a verdict on those algorithms; a comparison at matched budgets is required before any such claim, and it would

likely narrow the gap. The conclusions this paper draws about GWO–TS do not depend on those two rows, since GWO–TS runs at a smaller budget than the methods it is statistically level with on quality. Second, the grid-search protocol covered the four optimizers implemented in this study's codebase; the three remaining comparators were run at their reported configurations rather than re-tuned under an identical grid, so absolute margins may shift under a fully symmetric tuning effort. Third, the content-stage weights w_sim, w_pop and the diversification parameter lambda were fixed rather than swept, so the sensitivity of the satisfaction ranking, and hence of the candidate pool entering optimization at all, is unquantified. Fourth, travel times for non-car modes are estimates derived from a routing profile rather than measurements. A further caveat concerns aggregation: because fitness is not comparable in scale across trips of different length, the pooled statistics are meaningful only through the rank-based tests, which block by scenario; the pooled mean and standard deviation of fitness are not reported for that reason, and Table I is confined to a single scenario. None of these affects the principal finding, which concerns the relationship between quality and execution time: GWO–TS is statistically level with the strongest comparators on both solution quality and user satisfaction, and significantly faster than all of them.

Beyond tuning scope, four further limitations are inherited from the data pipeline. First, the final dataset contains 166 operational attractions and 181 hotels, which is small relative to Jakarta's true tourism inventory. This is a direct consequence of the deliberately conservative curation described in Section III.A: category whitelisting, keyword and name exclusion, business-status filtering and sub-venue consolidation each trade coverage for precision. The practical effect is that the candidate pool for any query is drawn from a comparatively narrow inventory, which favours spatially compact itineraries and may understate the combinatorial difficulty of the underlying problem; the results characterise the optimizers at this instance size and should not be extrapolated to a full metropolitan inventory without re-evaluation. Second, OSRM's public routing demo does not incorporate real-time traffic, so travel times do not reflect Jakarta's congestion patterns at specific times of day. Third, popularity signals from the Massive-STEPS check-in corpus span 2012–2018. Because popularity enters the satisfaction score with weight w_pop = 0.4, this historical signal materially influences which attractions reach the candidate pool at all: venues opened after 2018 carry no check-in evidence and are ranked on similarity and rating alone, while venues popular within that window but since declined may be over-weighted. The business-status filter removes venues closed at collection time, but it cannot correct for drift in relative popularity, and refreshing this signal from a contemporary check-in or mobility source is a prerequisite for deployment at scale. Fourth, the K-Means spatial zones used for the inter-zone penalty are a geometric clustering over coordinates and do not correspond to Jakarta's administrative district boundaries, so a "zone" in the fitness function may not match a tourist's intuitive notion of a neighborhood.

V. CONCLUSION

The authors presented an end-to-end, deployed multi-day tourist itinerary recommender for Jakarta that integrates machine-learning-based preference modeling with metaheuristic route optimization, addressing three gaps: multi-day planning (G1), user satisfaction as an explicit objective (G2), and a working ML-plus-optimization implementation (G3). At its core, GWO–TS adapts the Grey Wolf Optimizer to the permutation space with swap sequences and intensifies the incumbent leader every iteration with Tabu Search. Across 630 solver executions spanning seven optimizers, three scenarios and 30 paired runs each, the evidence separates cleanly. On user satisfaction the seven methods are statistically indistinguishable (Friedman $p = 0.7854$), with GWO–TS holding the highest mean and the lowest variance. On penalty-adjusted fitness the methods do differ, and GWO–TS is statistically level with the four strongest while significantly ahead of the two weakest. On runtime GWO–TS is significantly faster than every competitor, completing a three-day plan in 3.37 s. The claim this supports is a favourable balance: quality that cannot be distinguished from the leading group, at a cost that is distinguishably lower. That is the property which governs usability in an interactive service, and it is a narrower claim than uniform superiority.

Future work follows directly from the limitations above. The immediate priorities are a symmetric per-competitor grid search covering the three comparators not tuned here, a systematic report across the full scenario grid so that robustness to trip length, preference profile and budget ceiling can be assessed, and a small-scale evaluation with real tourists to complement the model-internal satisfaction score with self-reported measures. Beyond that, the formulation admits richer constraints such as ticket prices, weather and crowd-aware timing, additional travel modes derived from transit data, a refreshed popularity signal from a contemporary mobility source, and the combination of the content model with collaborative or sequential signals from the check-in corpus to strengthen personalization.

Reproducibility. The model repository and web client, including the main gwo_ts.py model and experiment implementation, are publicly available at github.com/Hazjel/sistem-rekomendasi-destinasi-wisata-jakarta and github.com/Hazjel/web-wisata-jakarta. Regarding our product website called Jakarta Routes, it is also publicly available at https://web-wisata-jakarta.vercel.app/.

CONFLICT OF INTEREST

The authors declare that there are no relevant financial or non-financial competing interests to report regarding this work.

ACKNOWLEDGMENT

The authors thank Telkom University especially for CoE HUMIC Engineering for supporting this research and the maintainers of OpenStreetMap, Massive-STEPS, Google Places, Wikipedia, and OSRM for the open data that made the data possible.

REFERENCES

[1] S. Shen, Y. Zhou, Q. Lei, and Z. Wu, "A survey of the orienteering problem: model evolution, algorithmic advances, and future directions," Dec. 2025, [Online]. Available: http://arxiv.org/abs/2512.16865

[2] P. Jewpanya, P. Nuangpirom, S. Pitjamit, and W. Nakkiew, "Optimized Travel Itineraries: Combining Mandatory Visits and Personalized Activities," Algorithms, vol. 18, no. 2, Feb. 2025, doi: 10.3390/a18020110.

[3] Y. Ding, L. Zhang, C. Huang, and R. Ge, "Two-stage travel itinerary recommendation optimization model considering stochastic traffic time," Expert Syst. Appl., vol. 237, Mar. 2024, doi: 10.1016/j.eswa.2023.121536.

[4] L. Florido-Benítez and J. A. Coca-Stefaniak, "Towards a new generation of smart tourism cities–GenAI-enabled aerotainment," Cities, vol. 167, Dec. 2025, doi: 10.1016/j.cities.2025.106311.

[5] C. Almeida and B. Alturas, "Users' satisfaction evaluation based on ISO standards for tourism and travel mobile applications," 2025. doi: 10.1504/ijitm.2025.144132.

[6] A. Bolufé-Röhler and D. Tamayo-Vera, "Machine Learning for Enhancing Metaheuristics in Global Optimization: A Comprehensive Review," Sep. 01, 2025, Multidisciplinary Digital Publishing Institute (MDPI). doi: 10.3390/math13182909.

[7] Y. Zhou, K. Zhou, and S. Chen, "Context-Aware Point-of-Interest Recommendation Based on Similar User Clustering and Tensor Factorization," ISPRS Int. J. Geoinf., vol. 12, no. 4, Apr. 2023, doi: 10.3390/ijgi12040145.

[8] Z. Wang et al., "Diversity-aware Deep Ranking Network for Recommendation," in International Conference on Information and Knowledge Management, Proceedings, Association for Computing Machinery, Oct. 2023, pp. 2564–2573. doi: 10.1145/3583780.3614848.

[9] A. Jadon and A. Patil, "A Comprehensive Survey of Evaluation Techniques for Recommendation Systems," Jan. 2024, [Online]. Available: http://arxiv.org/abs/2312.16015

[10] R. R. Chandan et al., "Genetic Algorithm and Machine Learning," 2023, pp. 167–182. doi: 10.4018/978-1-6684-5656-9.ch009.

[11] A. Casas-Ordaz et al., "Particle swarm optimization: A survey of innovations over the last 10 years," Comput. Sci. Rev., vol. 60, May 2026, doi: 10.1016/j.cosrev.2026.100910.

[12] M. Dorigo, V. Maniezzo, and A. Colorni, "The Ant System: Optimization by a colony of cooperating agents," 1996. doi: 10.1109/3477.484436.

[13] S. Mirjalili and A. Lewis, "The Whale Optimization Algorithm," Advances in Engineering Software, vol. 95, pp. 51–67, May 2016, doi: 10.1016/j.advengsoft.2016.01.008.

[14] S. Kirkpatrick, ; C D Gelatt, and ; M P Vecchi, "Optimization by Simulated Annealing," 1983. doi: 10.1126/science.220.4598.671.

[15] M. Dorigo and T. Stützle, "Ant Colony Optimization: Overview and Recent Advances," 2009. doi: 10.1007/978-1-4419-1665-5_8.

[16] M. S. Uzer, "New Hybrid Approaches Based on Swarm-Based Metaheuristic Algorithms and Applications to Optimization Problems," Applied Sciences (Switzerland), vol. 15, no. 3, Feb. 2025, doi: 10.3390/app15031355.

[17] A. Angelov and M. Lazarova, "Hybrid Artificial Bee Colony Algorithm for Test Case Generation and Optimization," Algorithms, vol. 18, no. 10, Oct. 2025, doi: 10.3390/a18100668.

[18] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer," Advances in Engineering Software, vol. 69, pp. 46–61, 2014, doi: 10.1016/j.advengsoft.2013.12.007.

[19] M. H. Nadimi-Shahraki, S. Taghian, and S. Mirjalili, "An improved grey wolf optimizer for solving engineering problems," Expert Syst. Appl., vol. 166, Mar. 2021, doi: 10.1016/j.eswa.2020.113917.

[20] K. Panwar and K. Deep, "Discrete Grey Wolf Optimizer for symmetric travelling salesman problem," Appl. Soft Comput., vol. 105, Jul. 2021, doi: 10.1016/j.asoc.2021.107298.

[21] S. Szénási and G. Légrádi, "Machine learning aided metaheuristics: A comprehensive review of hybrid local search methods," Dec. 15, 2024, Elsevier Ltd. doi: 10.1016/j.eswa.2024.125192.

[22] H. Zhang, J. Yan, and L. Wang, "Hybrid Tabu-Grey wolf optimizer algorithm for enhancing fresh cold-chain logistics distribution," PLoS One, vol. 19, no. 8, Aug. 2024, doi: 10.1371/journal.pone.0306166.

[23] R. S. Ghoneim, M. Arabasy, R. A. Osbaa, and R. Al Hamad, "A Systematic Literature Review on Integrating Machine Learning Algorithms and Metaheuristic Algorithms in Optimizing Sustainable Digital Architectural Design," Civil Engineering and Architecture, vol. 13, no. 3, pp. 1913–1931, May 2025, doi: 10.13189/cea.2025.130334.

[24] Carbonell, J. and Goldstein, J., "The Use of MMR, Diversity-Based Reranking for Reordering Documents and Producing Summaries."doi: 10.1145/290941.291025.