IAES International Journal of Artificial Intelligence (IJ-AI)
Vol. 15, No. 2, April 2026, pp. 1261∼1274
ISSN: 2252-8938, DOI: 10.11591/ijai.v15.i2.pp1261-1274

Genetic algorithm for generalized time-window assignment problem

Ali Kansou[1], Bilal Kanso[1], Houssein Wehbe[1,2,3], Haydar Bazzi[1], Ali Mcheik[1]
[1]Department of Computer Science, Faculty of Sciences, Lebanese University, Beirut, Lebanon
[2]School of Arts and Sciences, Lebanese International University, Nabatieh, Lebanon
[3]College of Arts and Sciences, American University of Iraq-Baghdad, Baghdad, Iraq

Article Info

Article history:

Received Aug 1, 2025
Revised Jan 3, 2026
Accepted Jan 22, 2026

Keywords:

Assignment with time windows
Constructive heuristic
Genetic algorithms
Local search procedure
Multi-resource scheduling

ABSTRACT

This paper presents a hybrid genetic algorithm (GA) for the generalized time-window assignment problem (GTWAP), a complex artificial intelligence (AI) scheduling challenge that involves assigning agents to resources under strict temporal and capacity constraints. Our method integrates a problem-specific heuristics and a repair mechanism to generate feasible and high-quality solutions. We provide a mathematical formulation for GTWAP and introduce a new public benchmark set, using CPLEX to obtain exact solutions. Computational experiments demonstrate that our GA is highly competitive with CPLEX, often matching its performance. This effectiveness makes our method a practical and scalable AI-driven tool for complex scheduling in domains like logistics and healthcare.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Bilal Kanso
Department of Computer Science, Faculty of Sciences, Lebanese University
Beirut, Lebanon
Email: bilal_kanso@hotmail.com

1. INTRODUCTION

Assignment and scheduling problems have become increasingly complex in recent years due to the introduction of temporal constraints, posing significant challenges for artificial intelligence (AI) and operations research. A common challenge in many real-world applications is matching agents (e.g., patients, workers, or voters) to appropriate facilities (e.g., medical centers, job sites, and polling stations) while respecting facility capacities and operational time constraints. Traditional assignment models, such as the facility location problem or the classical linear assignment problem, often fail to capture the key aspects such as agents' temporal availability, spatial dispersion, and resource heterogeneity. To bridge this gap, we propose the generalized time-window assignment problem (GTWAP), a flexible and unified assignment framework capable of representing a wide range of real-world situations. Unlike domain-specific approaches, such as staff scheduling or vehicle routing with time windows, GTWAP integrates multiple factors, including eligibility assignment rules, facility capacities and time window constraints, into a single model. This allows the development of scalable and efficient algorithms that can be used in diverse fields such as public administration, healthcare, maintenance, and workforce management.

In GTWAP, facilities (e.g., polling stations, vaccination centers, or workstations) operate under strict capacity limits and specific opening hours, each with different service durations. Agents, in turn, have their own availability and preferences regarding which facilities they can access. The objective is to assign agents to suitable facilities while satisfying all constraints, making GTWAP a powerful tool for solving many assignment

problems used in different areas (healthcare, logistics, education, and workforce coordination). To address GTWAP, genetic algorithm (GA) that goes beyond standard implementations is proposed. Initial experiments showed that standard GA operators struggle under the problem's tight constraints, often generating infeasible solutions. To overcome this, our approach uses a structured chromosome representation aligned with decision variables, together with a specialized repair mechanism and constraint-handling strategies (best allocation procedure (BAP)). This mechanism ensures feasibility and is integrated directly into evolutionary process.

This paper makes several key contributions. First, we introduce a novel integer linear programming formulation for GTWAP that captures the essential constraints and objectives of a wide range of real-world scenarios. Second, we propose a constructive assignment heuristic to generate high-quality initial solutions. Third, we implement two metaheuristic approaches: a GA specifically designed for GTWAP, and a general variable neighborhood search (GVNS) used here mainly to evaluate the GA performance. For further assessment, we use ILOG CPLEX Optimizer to obtain high-quality solutions and lower bounds. Fourth, since no public benchmark exists for GTWAP, we develop a dedicated and diverse benchmark set that reflects realistic and varied assignment contexts. Computational experiments demonstrate that the GA consistently improves upon the initial heuristic, is competitive with CPLEX and shows a slight performance advantage over GVNS on larger instances. Its convergence is further validated using statistical tests, confirming its potential as a practical tool for large-scale scheduling problems.

The remainder of this paper is organized as follows. Section 2 introduces problem definition. Section 3 presents the mathematical formulation. Section 4 describes the GA-based solution and GVNS approach to solve GTWAP. Section 5 presents and discusses the computational results. Finally, section 6 presents the conclusion.

Assignment and scheduling problems with temporal and capacity constraints have been widely studied across various domains. These problems often integrate time windows, resource availability, and spatial dispersion of agents and facilities. Recent work has increasingly focused on assigning agents to service windows or tasks under tight temporal and capacity constraints. For example, Paradiso *et al.* [1] proposes a stochastic look-ahead method for dynamic window assignment under uncertainty. Cavaliere *et al*. [2] develops two-stage models for time-window assignment in traveling salesperson problem (TSP) settings with stochastic travel times. Demirbilek [3] investigates optional and mandatory assignment strategies in dynamic vehicle routing under hard time windows. Côté *et al*. [4] addresses multi-activity workforce scheduling by integrating sequencing and time-window constraints within a unified optimization framework, whereas Gozali [5] designs a modified GA with local search to handle time-constrained worker assignment in manufacturing environments.

These contributions show how assignment problems become increasingly complex when time windows, agent availability, and capacity constraints interact. At the same time, earlier studies have explored assignment and scheduling problems, offering key modeling frameworks and solution strategies. For example, Schmidt *et al.* [6] formulates an integer programming model to consolidate polling places while minimizing voter travel and ensuring acceptable wait times, whereas Durán *et al.* [7] develops an analytical optimization model to redesign voter assignments considering both geographic distance and queue delays. Similar modeling perspectives appear in workforce scheduling [8], course assignment [9], and health resource allocation [10] where time-dependent availability and heterogeneity complicate the decision process. Other works focus on generalized variants of the assignment problem [11], dynamic supply chains under uncertainty [12], or spatio-temporal task allocation in crowdsourcing and mobile sensing [13], [14]. These contributions demonstrate the diversity of application settings but typically emphasize model construction and exact optimization, which often face serious scalability limitations as problem size increases. To address these challenges, metaheuristic and hybrid approaches have become central to large-scale, time-constrained assignment problems. Early studies applied pure GAs with integer or random-key representations [15]–[17], showing that even standard operators can produce competitive solutions compared to exact solvers. However, these classical GAs often required long runtimes to converge, or converged quickly at the expense of diversity and robustness. To address this, heuristic augmentations have been proposed. For instance, Kotwal and Dhope [18] integrated domain-specific heuristics into GA decoding, while Younas *et al.* [19] compared GA with particle swarm optimization (PSO) and differential evolution (DE), demonstrating superior solution quality but higher computational overhead.

More recently, hybridization trends have intensified. Reinforcement learning has been coupled with GAs for adaptive task assignment [20], and machine learning has been used to enhance GA-based order picking under fatigue effects [21]. Such approaches confirm the benefits of embedding learning or heuristic guidance into evolutionary search to balance exploration and exploitation. Beyond single-objective GAs, advances in multi-objective optimization have demonstrated the effectiveness of evolutionary algorithms for

complex combinatorial problems. Enhanced NSGA-II variants [22], [23] and MOEA/D approaches [24], [25] have shown strong performance and excellent convergence in logistics and urban systems, while hyper-heuristics [26], [27] offer adaptive strategies to dynamically balance exploration and exploitation. Nevertheless, despite their success, applications of these methods to assignment problems with strict time windows and heterogeneous resource constraints remain limited. This gap calls for specialized algorithms, such as GA for GTWAP, that can directly handle feasibility challenges while maintaining computational efficiency.

2. PROBLEM DESCRIPTION

Consider a set V of N agents and a set F of M facilities. Each agent $i \in V$ has a subset of preferred facilities $P_i \subseteq F$. The service time window for agent i at facility $f \in P_i$ is denoted by $T_i^f = [a_i^f, b_i^f]$, where a_i^f and b_i^f represent the earliest and latest possible starting times, respectively. The actual starting time is denoted by ST_i^f and must fall within both the time windows T_i^f and $[A_f, B_f]$ associated with the facility f. Each facility f has a daily capacity M_{max}^f representing the maximum number of agents it can serve. As in real-world, agents may require different service durations S_i. Two asymmetric distances: d_{if} (from agent i to facility f) and d_{fi} (return distance) are considered. The total round-trip distance for agent i to facility f is denoted by D_i^f and equal to $d_{if} + d_{fi}$. Agents cannot arrive late; if they arrive early to the facility, they must wait until their scheduled time. Waiting times are therefore not considered in the cost function. The objective of our problem is to assign each agent $i \in V$ to exactly one of their preferred facilities and determine a feasible service start time such that all constraints are satisfied. The total distances traveled by all agents must be minimized, while respecting individual service durations, facility time windows, agent time windows, agent preference facilities list and facility capacity limits.

Table 1 illustrates an instance with $N = 8$ agents and $M = 3$ facilities. We assume that facilities operate from 8:00AM to 10:00AM (i.e. $A_f = 8$, $B_f = 10$ for all $f \in F$), and each facility can serve up to three agents per day (i.e. $M_{max}^f = 3$). Each agent has two preferred facilities. The table lists service duration S_i, preferred facilities P_i, total travel distance D_i^f, and time windows T_i^f. A feasible solution L consists of M ordered lists L_f, one for each facility $f \in F$, where L_f represents a sequence of agents assigned to f satisfying all the problem constraints. Figure 1 illustrates the problem and its solution. Specifically, Figure 1(a) presents an instance of the problem, while Figure 1(b) shows a corresponding feasible solution composed of three ordered lists. The feasible proposed solution is denoted by $L = \bigcup_{f \in F} L_f$. The total travel cost of the solution is computed as $cost(L) = \sum_{f \in F} cost(L_f) = cost(L_1) + cost(L_2) + cost(L_3) = 170 + 170 + 120 = 460$.

Table 1. Instance example with 8 agents and 3 facilities

Agents	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8
$S_i(min)$	5	10	5	5	10	5	5	5
P_i	f_1, f_2	f_2,f_3	f_1, f_3	f_2, f_3	f_1, f_2	f_1, f_3	f_2, f_3	f_1, f_2
$D_i^{f_1}$	60	∞	60	∞	50	60	∞	50
$D_i^{f_2}$	50	50	∞	60	60	∞	70	80
$D_i^{f_3}$	∞	50	80	70	∞	70	70	∞
$T_i^{f_1}$	[9-10]	[0-0]	[8h30-9h30]	[0-0]	[9-10]	[8-9]	[0-0]	[8-8h30]
$T_i^{f_2}$	[8-9]	[8-9]	[0-0]	[8h40-10]	[8h30-9h30]	[0-0]	[8-9]	[9-10]
$T_i^{f_3}$	[0-0]	[8-9]	[9-10]	[9-10]	[0-0]	[9-10]	[8-9]	[0-0]

Figure 1. Problem instance and its solution: (a) problem solution instance and (b) solution-lists representation

3. MATHEMATICAL MODEL

This problem introduces novel constraints particularly the preferred facility requirement, that distinguish it from classical scheduling and assignment problems in the literature. To formalize our problem, we adapt the mathematical models of the vehicle routing problem with time windows (VRPTW) [28]. We consider the following two decision variables: i) $x_{ij}^f = 1$ if agent j is served immediately after agent i at facility f, and 0 otherwise; ii) ST_i^f is the actual starting time of agent i if assigned to facility f. We assign a fictitious starting agent denoted by 0 to each facility $f \in F$ with $T_0^f = [A_f, B_f]$ and $D_0^f = 0$.

The problem is formulated as the following integer linear programming model: (1) is the objective function, which minimizes the total travel distance for all agents. Constraint (2) ensures that each agent is assigned to exactly one preferred facility $f \in P_i$. Constraint (3) defines the start and end of service at each facility. Constraint (4) ensures the connectivity constraints between agents. Constraints (5) to (7) ensure the respect of the time windows of both agents and facilities. Constraint (8) ensures that the limit of the number of agents per facility f does not exceed M_{max}^f. Constraints (9) and (10) ensure that every agent is served exactly once. Constraint (11) states that each agent can only be assigned to facilities in their preferred set P_i. Constraints (12) and (13) define the variable domains for all used decision variables.

Minimize:

$$\sum_{f \in F} \sum_{i \in V \cup \{0\}} \sum_{j \in V \cup \{0\}} x_{ij}^f D_j^f \tag{1}$$

Subject to:

$$\sum_{f \in P_i} \sum_{j \in V \cup \{0\}, j \neq i} x_{ij}^f = 1 \quad \forall i \in V \tag{2}$$

$$\sum_{j \in V} x_{0j}^f = \sum_{i \in V} x_{i0}^f = 1 \qquad \forall f \in F \tag{3}$$

$$\sum_{j \in V, j \neq i} x_{ij}^f = \sum_{j \in V, j \neq i} x_{ji}^f \qquad \forall f \in F, \forall i \in V \cup \{0\} \tag{4}$$

$$ST_i^f + S_i \times x_{ij}^f \leq ST_j^f + [1 - x_{ij}^f] b_i^f \qquad \forall i, j \in V \cup \{0\}, \forall f \in F \tag{5}$$

$$a_i^f \sum_{j \in V \cup \{0\}, j \neq i} x_{ij}^f \leq ST_i^f \leq b_i^f \sum_{j \in V \cup \{0\}, j \neq i} x_{ij}^f \qquad \forall i \in V, \forall f \in F \tag{6}$$

$$a_i^f \leq ST_i^f \leq b_i^f \qquad \forall f \in F, \forall i \in V \cup \{0\} \tag{7}$$

$$\sum_{i \in V \cup \{0\}} \sum_{j \in V \cup \{0\}} x_{ij}^f \leq M_{max}^f + 1 \qquad \forall f \in F \tag{8}$$

$$x_{ii}^f = 0 \qquad \forall f \in F, \forall i \in V \cup \{0\} \tag{9}$$

$$x_{ij}^f + x_{ji}^f \leq 1 \qquad \forall f \in F, \forall i \in V \cup \{0\}, \forall j \in V \cup \{0\} \tag{10}$$

$$\sum_{j \in V \cup \{0\}} x_{ij}^f = \sum_{k \in V \cup \{0\}} x_{ki}^f = 0 \qquad \forall i \in V, \forall f \notin P_i \tag{11}$$

$$x_{ij}^f \in \{0, 1\} \qquad \forall i, j \in V \cup \{0\}, \forall f \in F \tag{12}$$

$$ST_i^f \in \mathbb{R}^+ \qquad \forall i \in V \cup \{0\}, \forall f \in F \tag{13}$$

4. SOLUTION METHOD

This section presents two meta-heuristic approaches developed to solve studied problem. The first is GA specifically designed to handle the complex constraints of GTWAP. The second is GVNS method, which serves both as alternative solution approach and as means of validating the performance of primary method.

4.1. Genetic algorithm for the assignment problem

GAs, a class of metaheuristics inspired by the principles of natural selection and genetics, were first proposed in [29]. For the GTWAP, the critical step lies in designing a suitable chromosome representation that encodes a feasible solution. We adopt an appropriate representation scheme using an $(N + M)$-dimensional vector composed of M-lists of agents, as illustrated in Figure 1. Our GA implementation follows the framework of [30] and is structured as follows: i) construct an initial population of N_{pop} feasible candidate solutions; ii) rank the solutions of the current population in ascending order according to the 'fitness' cost, defined as the total distance traveled by all agents; iii) select two parent solutions for reproduction using an elitist selection scheme; iv) apply a crossover operator to the selected parents to generate children solutions; v) improve diversity by applying a mutation procedure; vi) form a new population by replacing solutions with low fitness cost with improved ones; vii) check the stopping criterion, defined by reaching the predefined maximum number of iterations (N_{iter}). In the following subsections, we provide a detailed explanation of each component of our proposed GA.

4.1.1. Initial population

The initial population contains N_{pop} feasible solutions and is built as follows. One of these solutions is generated using a dedicated heuristic called the best allocation agent (BAA) method, while the rest of the solutions are generated via a random allocation method. The BAA method builds a solution iteratively by assigning agents to facilities. At each step, it selects the best unassigned agent and assigns it to its most preferred facility, i.e., the facility that minimizes its allocation cost. If no agents remain unassigned, the feasible solution is found, and the process terminates. If no feasible allocation is possible, a repair procedure called the BAP is applied. This procedure consists of randomly destroying 2 or 3 lists in the current (non-feasible) solution and then trying to complete it by randomly reassigning agents to their best possible facilities. This procedure is repeated up to 5 times. If a feasible solution is obtained within these attempts, it is accepted. Otherwise, the BAA method is restarted. We apply this method 100 times, and the best feasible solution will be considered as part of the initial population. On the other hand, the random solutions are generated using a similar process, but with one key difference: the agent to be assigned at each iteration is selected randomly rather than optimally.

4.1.2. Selection mechanism

In this work, we use a fitness-based selection method to guide the evolution of the population. This step plays a critical role in the GA by selecting the fittest individuals from the current population to reproduce and create the next generation. Our selection process relies on two key principles: i) the best individuals of the population are directly preserved in the next population without modification, and ii) this ensures that the solutions with the highest fitness cost are never lost during reproduction.

4.1.3. One-point crossover operator

Once two parents are selected from the current population using the selection method, we apply our three-step crossover operator to generate children, as illustrated in Figure 2. First, we select randomly two agents v and w from two different random lists L_A in parent A and L_B in parent B (see Figure 2(a)). In the second step, we remove all agents following v in L_A and all agents preceding w in L_B, including v and w themselves (see Figure 2(b)). This yields two intermediate infeasible children, each missing the agents removed in this step. In the third step, we repair these infeasible solutions using the BAP, by reassigning missing agents to their most suitable facilities while satisfying time window and capacity constraints (see Figure 2(c)). The two resulting children are now feasible and can be evaluated further. Note that when this process fails to assign all agents to facilities, we use the destruction-construction procedure (BAP) as part of the repair process for the children. Figure 2 illustrates the three steps of the crossover process and the resulting children. As shown, the two resulting children outperform their parents: the second parent has a total cost of 510, while its child achieves a better cost of 490, representing a very good improvement of 20. Finally, each child undergoes a further improvement as mutation, which is detailed in the next section.

4.1.4. Mutation strategy

In this phase of GA, we enhance the feasible solutions generated by one-point crossover via a mutation procedure applied with a predefined probability. The mutation consists of a sequence of local search procedures applied up to N_M times and in the following order: relocate, swap, then BAP. Figures 3 and 4 illustrate how the relocate and swap procedures work using the same example presented in section 2 (Table 1). Figure 3(a) shows the initial solution, while Figure 3(b) presents the result after applying the relocate move: agent v_4 is reassigned, yielding a cost improvement of 10. Similarly, Figure 4(a) depicts the solution before the while swap move, and Figure 4(b) shows the solution after swapping agents v_1 and v_8, resulting in a more significant improvement of 40. Note, that after applying the mutation, both the improved children and their parent solutions are gathered into a single list, which is then arranged in ascending order based on the total solution cost.

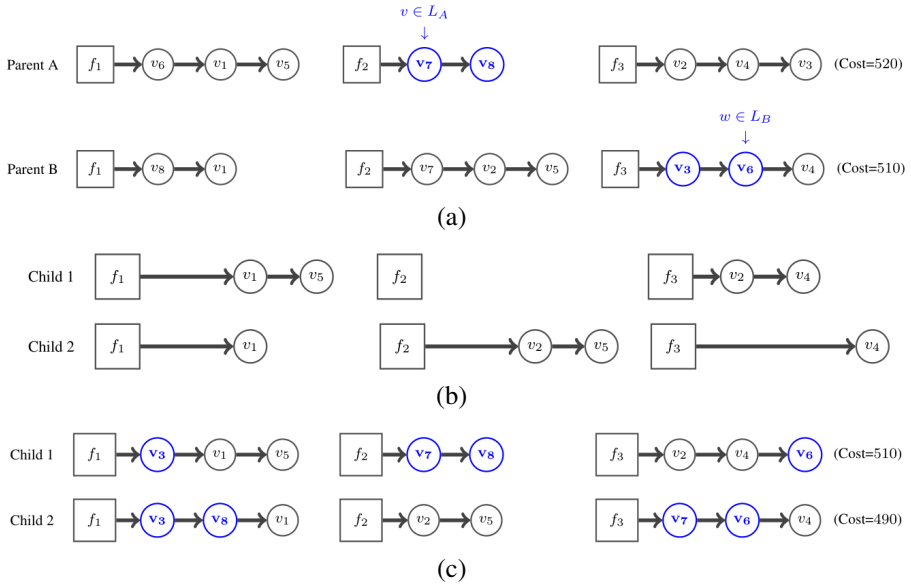

Figure 2. Illustration of the three steps of the crossover process: (a) agent selection, (b) removal and infeasible child generation, and (c) repair via BAP to ensure feasibility

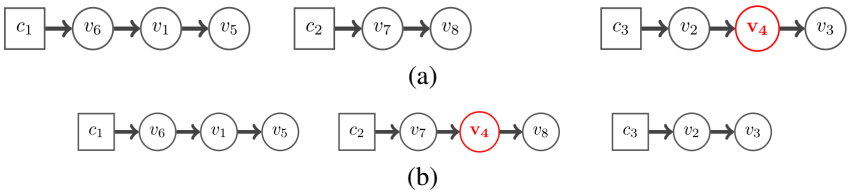

Figure 3. Illustration of the relocate procedure (a) before relocate: v_4 assigned to L_3 with a total cost of 520 and (b) after relocate: v_4 moved to L_2, reducing the total cost from 10 to 510

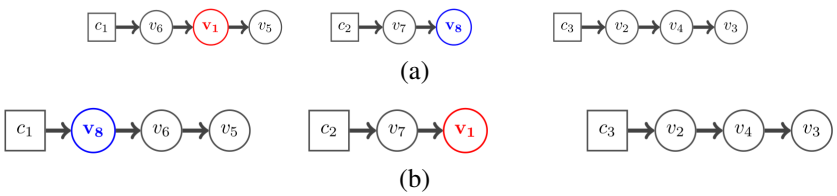

Figure 4. Illustration of the swap procedure (a) before swap: v_1 assigned to L_1 and v_8 to L_2, with a total cost of 520 and (b) after swap: v_1 and v_8 exchange their facilities, reducing the total cost from 520 to 480

4.2. General variable neighborhood search approach

To validate our GA and provide an alternative solution method, we developed an algorithm based on a GVNS. GVNS is a powerful metaheuristic used for combinatorial optimization problems, that changes systematically neighborhood structures to escape local optima [31]. The idea of this metaheuristic is to alternate between two phases at each iteration: stochastic shaking and local search. The shaking (or perturbation) phase is used to perturb the current solution by finding a neighborhood solution S' after applying in sequence one move from the three moves (relocate, swap, and BAP).

The local search then refines the perturbed solution by exploring improvements within its immediate neighborhood. Whenever a neighborhood yields an improvement, GVNS updates the new current solution and resets the search with the smallest neighborhood. Otherwise, it proceeds to the next neighborhood. This process is repeated for a predefined number of iterations (β_{max}) to allow a robust exploration of the solution space. The pseudo-code of the GVNS algorithm is explained in Algorithm 1.

Algorithm 1: The pseudo code of GVNS algorithm

```
Input: an initial solution S_0, the maximum number of iterations β_max = 2000
Initialization: iter = 1, S = S_0
while iter ≤ β_max do
    k=1
    while k ≤ 3 do
        Apply k − th shaking move to obtain S'
        Apply local search phase to find a solution S'*
        If cost(S'*) < cost(S)
            then S ← S'*
                 k = 1
        Else
        k=k+1
    iter = iter + 1
```

5. EXPERIMENTS AND RESULTS

This section presents the results obtained using the IBM ILOG CPLEX CP Optimizer and our proposed GA and GVNS methods for solving the GTWAP. The mathematical model was implemented in CPLEX with the CP Optimizer solver, which was used to compute either optimal solutions or valid lower bounds. CP Optimizer [32] is a state-of-the-art constraint programming engine that extends classical CP with advanced scheduling constructs. Its exact search algorithm integrates metaheuristics, such as self-adaptive large neighborhood search, enabling fast discovery of high-quality solutions and, in many cases, proofs of optimality.

5.1. Benchmark description

To evaluate the performance of our methods, we developed a dedicated GTWAP benchmark. This benchmark includes instances of different sizes and characteristics, designed to reflect realistic and diverse operational scenarios. The number of agents ranges from 20 to 1187, while the number of available facilities varies from 2 to 20. Agent time windows differ in width, ranging from tight to wide or even absent, allowing evaluation under different levels of scheduling flexibility. Assignment durations depend on the facility and are chosen to approximate the minimal gap between time windows, with some controlled random variations.

Each agent must be assigned exactly once, and the total number of available resources for all facilities matches the number of agents. Facility time windows are randomly distributed, and agent arrival and service times are aligned with the overall problem parameters. These instances were randomly generated according to some criteria such as normal distribution and clustering. Service durations vary by facility and roughly correspond to the minimum time window differences, with additional random adjustments. The benchmark comprises 130 instances divided into two subsets: 65 smaller to medium-sized instances with the number of agents ranges from 20 to 98, and facilities from 2 to 8, and 65 larger and more complex instances with 144 to 1187 agents and 4 to 20 facilities. This set is used to test scalability and robustness of the methods.

5.2. Results

Table 2 summarizes the results for the first set. The columns correspond respectively to: i) the instance name; ii) the number of agents; iii) the number of facilities; iv) the lower bound (lb) computed by the CP

Optimizer solver; v) the cost obtained during the creation of the initial population; vi) the cost obtained by our GA; vii) the percentage gap between the GA cost and the lower bound (GAP1); viii) the percentage improvement of GA over the initial method (GAP2); and ix) the number of feasible solutions found by the CP solver within the time limit. The results are presented as listed in the table.

Table 2. Results on the first set of instances

Instance	# Agents	# Facilities	LB	INIT	GA	GAP1 (%)	GAP2 (%)	# Solutions
gtwap1	20	2	**372**	428	372	0	13.08	11
gtwap2	20	2	**450**	482	450	0	6.64	12
gtwap3	20	2	**490**	514	490	0	4.67	7
gtwap4	20	2	**430**	446	430	0	3.59	8
gtwap5	20	2	**468**	494	468	0	5.26	10
gtwap6	20	2	**480**	512	480	0	6.25	10
gtwap7	20	2	**482**	508	482	0	5.12	9
gtwap8	20	2	**514**	530	514	0	3.02	10
gtwap9	20	2	**376**	434	376	0	13.36	13
gtwap10	20	2	**506**	532	506	0	4.89	7
gtwap11	20	2	**498**	538	498	0	7.43	10
gtwap12	20	2	**462**	510	462	0	9.41	15
gtwap13	20	2	**498**	546	498	0	8.79	7
gtwap14	20	2	**478**	512	478	0	6.64	9
gtwap15	20	2	**492**	512	492	0	3.91	5
gtwap16	20	2	**606**	652	606	0	7.06	10
gtwap17	20	2	**468**	492	468	0	4.88	19
gtwap18	20	2	**582**	610	582	0	4.59	7
gtwap19	20	2	**480**	516	480	0	6.98	8
gtwap20	20	2	**538**	580	538	0	7.24	10
gtwap21	20	2	**396**	418	396	0	5.26	7
gtwap22	20	2	**352**	380	352	0	7.37	7
gtwap23	20	2	**388**	432	388	0	10.19	13
gtwap24	20	2	**412**	444	412	0	7.21	8
gtwap25	20	2	**336**	380	336	0	11.58	13
gtwap26	48	4	2470	2658	2470	0	7.07	47
gtwap27	50	5	928	1010	928	0	8.12	29
gtwap28	50	5	1084	1144	1084	0	5.24	39
gtwap29	50	5	976	1064	976	0	8.27	34
gtwap30	50	5	974	1048	974	0	7.06	27
gtwap31	50	5	1072	1152	1072	0	6.94	35
gtwap32	50	5	868	958	868	0	9.39	24
gtwap33	50	5	826	928	826	0	10.99	32
gtwap34	50	5	904	962	904	0	6.03	37
gtwap35	50	5	900	950	900	0	5.26	36
gtwap36	50	5	926	1016	926	0	8.86	35
gtwap37	50	5	558	640	558	0	12.81	17
gtwap38	50	5	546	624	546	0	12.5	31
gtwap39	50	5	462	534	462	0	13.48	34
gtwap40	50	5	468	520	468	0	10	39
gtwap41	50	5	442	526	442	0	15.97	31
gtwap42	72	6	3314	3694	3314	0	10.29	71
gtwap43	80	8	1322	1436	1322	0	7.94	52
gtwap44	80	8	1306	1430	1306	0	8.67	62
gtwap45	80	8	1396	1522	1396	0	8.28	80
gtwap46	80	8	1384	1508	1384	0	8.22	75
gtwap47	80	8	1466	1570	1466	0	6.62	81
gtwap48	80	8	998	1078	998	0	7.42	45
gtwap49	80	8	1072	1200	1072	0	10.67	76
gtwap50	80	8	1092	1186	1092	0	7.93	80
gtwap51	80	8	1058	1180	1058	0	10.34	69
gtwap52	80	8	1006	1116	1006	0	9.86	74
gtwap53	86	8	1334	1402	1332	-0.15	4.99	18
gtwap54	86	8	1437	1466	1437	0	1.98	23
gtwap55	86	8	1418	1465	1422	0.28	2.94	44
gtwap56	86	8	1297	1301	1297	0	0.31	22
gtwap57	90	7	1525	1604	1530	0.33	4.61	11
gtwap58	90	7	1550	1599	1556	0.39	2.69	32
gtwap59	96	4	4968	5166	4962	-0.12	3.95	93
gtwap60	96	4	4978	5220	4962	-0.32	4.94	105
gtwap61	98	7	1756	1781	1752	-0.23	1.63	34
gtwap62	98	7	1647	1649	1647	0	0.12	65
gtwap63	98	7	1688	1702	1688	0	0.82	23
gtwap64	98	7	1691	1709	1684	-0.42	1.46	12
gtwap65	98	7	1778	1801	1789	0.61	0.67	31

Solutions proven to be optimal within the time limit are marked in bold. Note that for this set, GVNS and GA obtained the same results, thus, only GA results are reported. Table 3 reports the results for all instances in the second set. The columns correspond respectively to: instance name, number of agents, number of facilities, INIT cost, GA cost, GVNS cost, percentage gap of GA over INIT and percentage gap of GVNS over INIT. Gaps are computed as: $\text{GAP} = \frac{\text{INIT - X}}{\text{INIT}} \times 100$ where $\text{X} \in \{\text{GA}, \text{GVNS}\}$.

Table 3. Results on the second set of instances

Instance	#Agents	#Facilities	INIT	GA	GVNS	GAP1 (%)	GAP2 (%)
gtwap66	144	4	8312	8168	8168	1.732	1.732
gtwap67	144	6	7532	7248	7388	3.771	1.912
gtwap68	144	4	8458	8168	8168	3.429	3.429
gtwap69	144	6	7494	7248	7248	3.283	3.283
gtwap70	148	4	2766	2470	2479	10.701	10.376
gtwap71	192	4	11016	10604	10604	3.740	3.740
gtwap72	192	4	10844	10604	10604	2.213	2.213
gtwap73	216	6	10454	10174	10189	2.678	2.535
gtwap74	216	6	10552	10174	10174	3.582	3.582
gtwap75	230	6	4129	3997	4001	3.197	3.1
gtwap76	238	6	4146	4130	4130	0.386	0.386
gtwap77	240	4	12748	12628	12628	0.941	0.941
gtwap78	240	4	12806	12628	12628	1.390	1.390
gtwap79	270	6	14928	14925	14925	0.020	0.020
gtwap80	274	6	4942	4933	4933	0.182	0.182
gtwap81	275	6	15321	14874	14874	2.918	2.918
gtwap82	288	4	15182	14648	14648	3.517	3.517
gtwap83	288	6	15230	15004	15045	1.484	1.215
gtwap84	288	4	14690	14562	14562	0.871	0.871
gtwap85	288	4	3738	3314	3333	11.343	11.835
gtwap86	288	6	16070	15016	15016	6.559	6.559
gtwap87	305	6	15961	15650	15650	1.948	1.948
gtwap88	305	6	5565	5520	5520	0.809	0.809
gtwap89	305	6	15438	15412	15412	0.168	0.168
gtwap90	355	6	16851	16327	16368	3.110	2.866
gtwap91	355	6	36392	36278	36278	0.313	0.313
gtwap92	360	4	46840	46390	46390	0.961	0.961
gtwap93	360	4	41490	40806	40806	1.649	1.649
gtwap94	360	6	43500	42268	42290	2.832	2.782
gtwap95	360	6	40388	38868	38868	3.763	3.763
gtwap96	400	6	6986	6971	6971	0.215	0.215
gtwap97	420	12	40458	39044	39051	3.495	3.478
gtwap98	420	12	40592	39050	39074	3.799	3.74
gtwap99	480	4	61712	61348	61348	0.590	0.590
gtwap100	480	4	54426	54100	54100	0.599	0.599
gtwap101	520	6	62196	61382	61382	1.309	1.309
gtwap102	520	6	56628	55412	55412	2.147	2.147
gtwap103	600	4	76516	76268	76268	0.324	0.324
gtwap104	600	4	69814	68696	68696	1.601	1.601
gtwap105	600	12	59002	57094	57094	3.234	3.234
gtwap106	600	12	60180	57238	57249	4.889	4.87
gtwap107	700	6	86586	83320	83320	3.772	3.772
gtwap108	700	6	79388	75364	7541	5.069	4.864
gtwap109	720	4	92182	91236	91236	1.026	1.026
gtwap110	720	4	82722	82344	82344	0.457	0.457
gtwap111	780	12	80570	77180	77186	4.208	4.2
gtwap112	780	12	80586	76868	76873	4.614	4.608
gtwap113	800	12	11109	10966	10966	1.287	1.287
gtwap114	800	12	20483	20260	20260	1.089	1.089
gtwap115	800	12	10761	10713	10713	0.446	0.446
gtwap116	800	12	11097	10989	10989	0.973	0.973
gtwap117	800	12	10593	10452	10452	1.331	1.331
gtwap118	829	15	10310	10230	10230	0.776	0.776
gtwap119	840	4	106144	105816	105816	0.309	0.309
gtwap120	840	4	97504	96904	96904	0.615	0.615
gtwap121	850	15	10215	10134	10134	0.793	0.793
gtwap122	880	6	95676	94388	94388	1.346	1.346
gtwap123	880	6	96172	94532	94541	1.705	1.696
gtwap124	960	4	120192	119910	119910	0.235	0.235
gtwap125	960	4	111514	110934	110934	0.520	0.520
gtwap126	960	12	94840	93516	93519	1.396	1.393
gtwap127	960	12	97080	93193	93220	4.004	3.976
gtwap128	1015	18	11596	11556	11556	0.345	0.345
gtwap129	1063	20	12221	12146	12146	0.614	0.614
gtwap130	1115	16	33873	33773	33773	0.295	0.295

5.3. Analysis and discussion

The INIT method was run for 1500 iterations, and each instance was solved five times; the reported values correspond to the average. For the first set, the CP Optimizer computed lower bounds with a time limit of three hours. In some cases, it found optimal solutions in under one minute. GA outperformed the CP Optimizer in 5 instances, while the CP outperformed GA in 4 instances. For the remaining 56 instances, both performed equally. The average gap between GA and CP Optimizer was only 0.005692%, and both methods solved 26 instances to optimality. This is expected, as the search space for small instances is relatively limited. The average number of feasible solutions per instances in the first set was approximately 31, making exhaustive exploration feasible within short computation times.

Compared to INIT, GA improved results by an average of 6.919% in Set 1 and 2.2% in Set 2, leading to an overall average gain of 4.56%. The execution time of the INIT method is very short, averaging less than 2 seconds for Set 1 and under 30 seconds for Set 2. As a result, INIT execution times were not reported in the tables. GA also ran efficiently, with average execution times of 10 seconds for Set 1 and 77 seconds for Set 2. GA and GVNS have the same performance on all 65 small instances. For the larger and more challenging instances in Set 2, GA demonstrated slightly better, outperforming GVNS in 16 instances. The gaps compared to INIT are 2.198% for GA and 2.139% for GVNS. Figure 5 graphically illustrates the results for INIT, GA and GVNS. Runtime for both methods are comparable as shown in Figure 6, making GA the preferred choice due to its marginally higher solution quality.

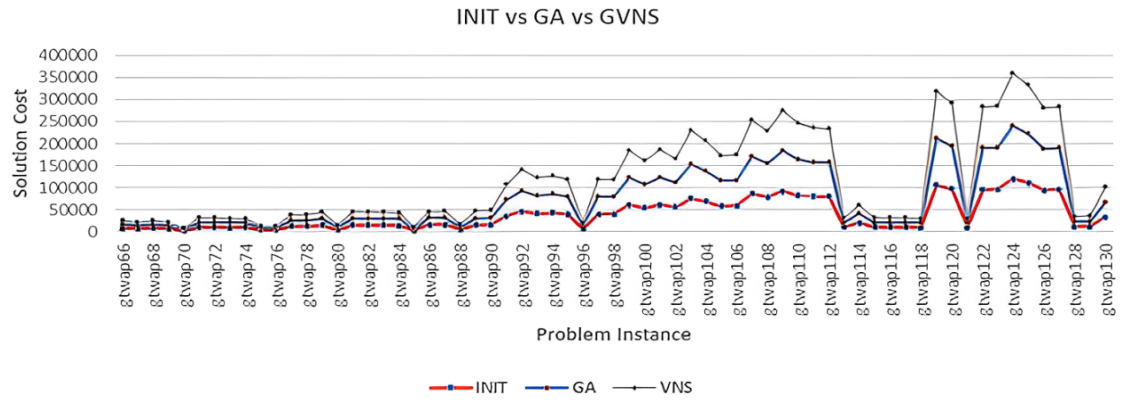

Figure 5. INIT vs GA vs GVNS

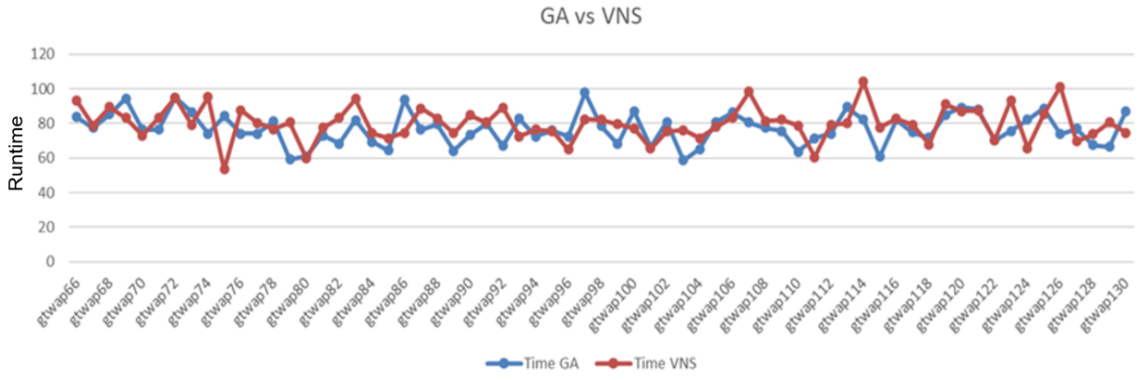

Figure 6. Time GA vs GVNS

To assess the effectiveness of our proposed approach, we applied the Wilcoxon signed-rank test, a widely used non-parametric statistical method to compare paired results between two different methods [33]. This test was applied to statistically evaluate the performance of the GA method relative to the INIT method across the two benchmark sets. The analysis, performed separately for each set, yielded p-values of 2.3×10^{-12}

and 2.405×10^{-12}. These p-values are extremely small below the commonly accepted thresholds of 0.05 or 0.01. In both cases, the sum of the signed ranks was zero, indicating that the GA method consistently outperformed the INIT method across all instances. These results provide strong statistical evidence that the differences observed between INIT and GA are not due to random variation, confirming the statistical significance of GA's superiority over INIT.

To further illustrate the performance of the GA, we analyzed its convergence behavior across the benchmark instances. Figure 7 shows the average best solution over 100 iterations, with shaded areas representing the min–max range across instances. The results reveal a steep improvement during the first 20–30 iterations, followed by a slower refinement phase, that indicates rapid early progress and stable convergence afterward. The narrowing of the min–max band demonstrates that the algorithm behaves consistently across different instances. Figure 8 depicts a heatmap that shows how the number of facilities and centers affects the GAP between INIT and GA solutions. Facilities arranged on the vertical axis and centers the horizontal axis. The color scale, ranging from dark purple (low GAP) to bright yellow (high GAP), reveals significant variation in performance.

Overall, the number of facilities appears to have a stronger impact than the number of centers. Larger instances with more facilities tend to have lower GAPs. Conversely, instances with a small to moderate number of facilities and a low number of centers show higher GAP values, suggesting that these instances are more challenging for the GA. This emphasizes how problem size and structure influence the algorithm's ability to close the gap between initial and optimized solutions.

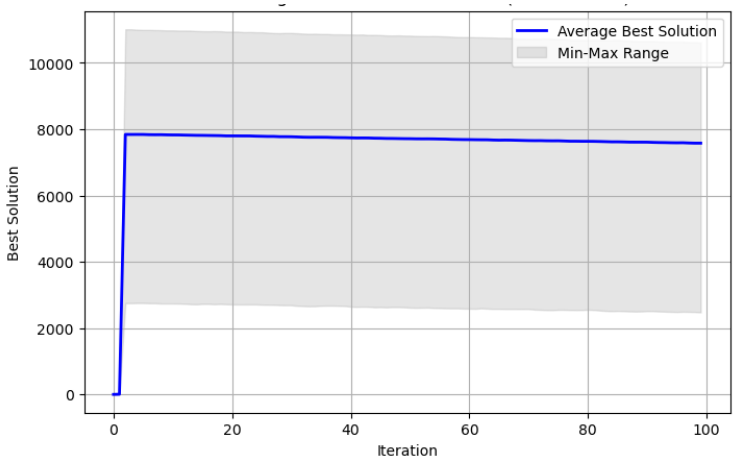

Figure 7. GA convergence over 100 iterations for Set 2

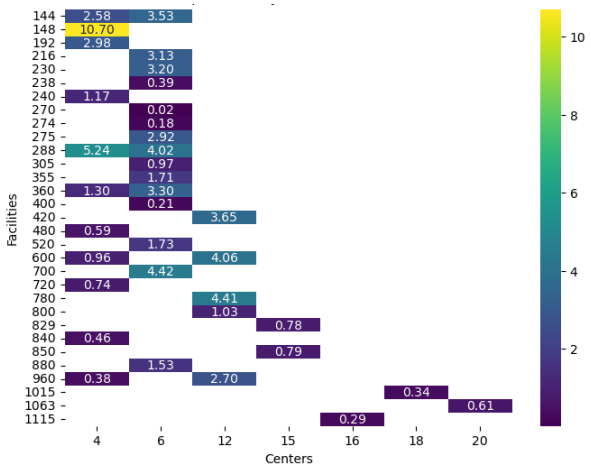

Figure 8. Effect of the number of facilities and centers on the GAP values

5.4. Settings

The results in Tables 2 and 3 were obtained after executing five runs of the algorithms GA and GVNS for each instance. The number of attempts to find an initial feasible solution using the INIT method was set to 1000. In GA, the population size N_{pop} was tested between 20 and 30 and the number of iterations is equal to 100. In GVNS, the neighborhood structures and shaking mechanisms were set according to the standard configuration used in previous studies, with a maximum of 1000 iterations per run. We selected these parameters when reaching the best values that obtain an improvement solutions with an acceptable computational time. All experiments were conducted on a machine equipped with a 2.6 GHz dual-core CPU, 32 GB of RAM, and Windows 11. Both GA and GVNS were implemented in Java.

6. CONCLUSION

In this work, we addressed the GTWAP by proposing a new mixed integer programming (MIP) formulation, a dedicated benchmark, and efficient solution methods, including a constructive heuristic and a GA-based metaheuristic. Experimental results show that the proposed GA consistently outperforms the heuristic, achieving good results within reasonable computational times. For future research, promising directions include hybridizing the GA with learning-based components, developing adaptive operators and memetic strategies to improve exploration-exploitation trade-offs, and framing GTWAP as a sequential decision-making task to enable reinforcement learning approaches and more autonomous scheduling systems.

FUNDING INFORMATION

This work has been co-funded by the Lebanese University and the Lebanese International University.

AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Ali Kansou	✓	✓	✓	✓	✓	✓		✓	✓	✓				✓
Bilal Kanso		✓				✓		✓	✓	✓	✓			✓
Houssein Wehbe	✓		✓	✓		✓			✓	✓	✓			✓
Haydar Bazzi	✓	✓	✓	✓		✓			✓	✓	✓			
Ali Mcheik	✓		✓	✓		✓	✓		✓	✓	✓			

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal Analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject Administration
Fu : **Fu**nding Acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

The data supporting the findings of this study are available from the corresponding author upon reasonable request.

REFERENCES

[1] R. Paradiso, R. Roberti, and M. Ulmer, "Lookahead scenario relaxation for dynamic time window assignment in service routing," *Transportation Research Part B: Methodological*, vol. 192, Feb. 2025, doi: 10.1016/j.trb.2024.103137.

[2] F. Cavaliere, M. Fischetti, R. Roberti, and D. Salvagnin, "Models and algorithms for the time window assignment traveling salesperson problem with stochastic travel times," *European Journal of Operational Research*, vol. 329, no. 1, pp. 96–111, Feb. 2026, doi: 10.1016/j.ejor.2025.07.034.
[3] M. Demirbilek, "Optional and mandatory assignment strategies for dynamic vehicle routing with time windows," *Ain Shams Engineering Journal*, vol. 16, no. 9, Sep. 2025, doi: 10.1016/j.asej.2025.103462.
[4] J.-F. Côté, R. Mansini, and A. Raffaele, "Multi-period time window assignment for attended home delivery," *European Journal of Operational Research*, vol. 316, no. 1, pp. 295–309, Jul. 2024, doi: 10.1016/j.ejor.2024.01.021.
[5] A. A. Gozali, "Modified genetic algorithm to solve worker assignment problem with time windows," *Industrial Artificial Intelligence*, vol. 2, no. 1, Feb. 2024, doi: 10.1007/s44244-024-00015-9.
[6] A. P. Schmidt, D. Buell, and L. A. Albert, "Optimal consolidation of polling locations," *Manufacturing & Service Operations Management*, vol. 26, no. 3, pp. 1028–1042, May 2024, doi: 10.1287/msom.2022.0497.
[7] G. Durán, M. Giormenti, M. Guajardo, P. M. Pinto, P. A. Rey, and N. E. S. Moses, "Improving access to voting with optimized matchings," *Electoral Studies*, vol. 51, pp. 38–48, Feb. 2018, doi: 10.1016/j.electstud.2017.11.004.
[8] P. Sun, L. P. Veelenturf, M. Hewitt, and T. V. Woensel, "Adaptive large neighborhood search for the time-dependent profitable pickup and delivery problem with time windows," *Transportation Research Part E: Logistics and Transportation Review*, vol. 138, Jun. 2020, doi: 10.1016/j.tre.2020.101942.
[9] Y. Chen, M. Bayanati, M. Ebrahimi, and S. Khalijian, "A novel optimization approach for educational class scheduling with considering the students and teachers' preferences," *Discrete Dynamics in Nature and Society*, vol. 2022, no. 1, Jan. 2022, doi: 10.1155/2022/5505631.
[10] R. P. Buvaneswari, S. M. Kumar, R. K. Kumar, S. M. Kumar, S. Rajeshwari, K. Karunambiga, "Optimizing healthcare resource allocation using residual convolutional neural networks," *Informing Science: The International Journal of an Emerging Transdiscipline*, vol. 28, 2025, doi: 10.28945/5449.
[11] W. Li and J. Ou, "The generalized assignment problem with fixed processing times and uniform processing costs to minimize total cost," *European Journal of Operational Research*, vol. 327, no. 2, pp. 420–431, Dec. 2025, doi: 10.1016/j.ejor.2025.05.031.
[12] R. Aldrighetti, D. Battini, D. Ivanov, and I. Zennaro, "Costs of resilience and disruptions in supply chain network design models: A review and future research directions," *International Journal of Production Economics*, vol. 235, May 2021, doi: 10.1016/j.ijpe.2021.108103.
[13] J. Chen *et al*., "DATA-WA: demand-based adaptive task assignment with dynamic worker availability windows," in *2025 IEEE 41st International Conference on Data Engineering (ICDE)*, 2025, pp. 850-862, doi: 10.1109/ICDE65448.2025.00069.
[14] S. Peng, K. Liu, S. Wang, Y. Xiang, B. Zhang, C. Li, "Time window-based online task assignment in mobile crowdsensing: problems and algorithms," *Peer-to-Peer Networking and Applications*, vol. 16, no. 2, pp. 1069–1087, Mar. 2023, doi: 10.1007/s12083-023-01454-4.
[15] A. Savić, D. Tošić, M. Marić, and J. Kratica, "Genetic algorithm approach for solving the task assignment problem," *Serdica Journal of Computing*, vol. 2, no. 3, pp. 267–276, Dec. 2008, doi: 10.55630/sjc.2008.2.267-276.
[16] J. Kratica, V. K. Vujcic, and M. Cangalovic, "Computing strong metric dimension of some special classes of graphs by genetic algorithms," *Yugoslav Journal of Operations Research*, vol. 18, no. 2, pp. 143–151, 2008, doi: 10.2298/YJOR0802143K.
[17] J. J. M. Mendes, J. F. Gonçalves, and M. G. C. Resende, "A random key based genetic algorithm for the resource constrained project scheduling problem," *Computers & Operations Research*, vol. 36, no. 1, pp. 92–109, Jan. 2009, doi: 10.1016/j.cor.2007.07.001.
[18] J. G. Kotwal and T. S. Dhope, "Solving task allocation to the worker using genetic algorithm," *International Journal of Computer Science and Information Technologies*, vol. 6, no. 4, pp. 3736–3741, 2015.
[19] I. Younas, F. Kamrani, M. Bashir, and J. Schubert, "Efficient genetic algorithms for optimal assignment of tasks to teams of agents," *Neurocomputing*, vol. 314, pp. 409–428, Nov. 2018, doi: 10.1016/j.neucom.2018.07.008.
[20] Z. Fang, T. Ma, J. Huang, Z. Niu, and F. Yang, "Efficient task allocation in multi-agent systems using reinforcement learning and genetic algorithm," *Applied Sciences*, vol. 15, no. 4, Feb. 2025, doi: 10.3390/app15041905.
[21] M. Gabellini, F. Calabrese, A. Regattieri, D. Loske, and M. Klumpp, "A hybrid approach integrating genetic algorithm and machine learning to solve the order picking batch assignment problem considering learning and fatigue of pickers," *Computers & Industrial Engineering*, vol. 191, May 2024, doi: 10.1016/j.cie.2024.110175.
[22] B. Doerr, T. Ivan, and M. S. Krejca, "Speeding up the NSGA-II with a simple tie-breaking rule," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 39, no. 25, pp. 26964–26972, Apr. 2025, doi: 10.1609/aaai.v39i25.34902.
[23] Q. Xiong *et al*., "Enhanced NSGA-II algorithm based on novel hybrid crossover operator to optimise water supply and ecology of Fenhe reservoir operation," *Scientific Reports*, vol. 14, no. 1, Dec. 2024, doi: 10.1038/s41598-024-80419-w.
[24] K. Javan *et al*., "Coupled SWMM-MOEA/D for multi-objective optimization of low impact development in urban stormwater systems," *Journal of Hydrology*, vol. 656, Aug. 2025, doi: 10.1016/j.jhydrol.2025.133044.
[25] C. Legrand, D. Cattaruzza, L. Jourdan, and M. Kessaci, "Improving neighborhood exploration into MOEA/D framework to solve a bi-objective routing problem," *International Transactions in Operational Research*, vol. 32, no. 1, pp. 117–143, Jan. 2025, doi: 10.1111/itor.13373.
[26] Z.-Q. Zhang, B. Qian, R. Hu, and J.-B. Yang, "Q-learning-based hyper-heuristic evolutionary algorithm for the distributed assembly blocking flowshop scheduling problem," *Applied Soft Computing*, vol. 146, Oct. 2023, doi: 10.1016/j.asoc.2023.110695.
[27] W. Bouazza, "Machine learning-based hyper-heuristics: a clear insight," in *Proceedings of the 2024 7th International Conference on Computational Intelligence and Intelligent Systems*, Nagoya Japan: ACM, Nov. 2024, pp. 29–37, doi: 10.1145/3708778.3708783.
[28] G. Laporte, "The vehicle routing problem: an overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 3, pp. 345–358, Jun. 1992, doi: 10.1016/0377-2217(92)90192-C.
[29] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. Cambridge, United States: The MIT Press, 1992, doi: 10.7551/mitpress/1090.001.0001.
[30] P. C. Chu and J. E. Beasley, "A genetic algorithm for the generalised assignment problem," *Computers & Operations Research*, vol. 24, no. 1, pp. 17–23, Jan. 1997, doi: 10.1016/S0305-0548(96)00032-9.
[31] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers & Operations Research*, vol. 24, no. 11, pp. 1097–1100, Nov. 1997, doi: 10.1016/S0305-0548(97)00031-2.

[32] P. Laborie, J. Rogerie, P. Shaw, and P. Vilím, "IBM ILOG CP optimizer for scheduling," *Constraints*, vol. 23, no. 2, pp. 210–250, Apr. 2018, doi: 10.1007/s10601-018-9281-x.
[33] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization," *Journal of Heuristics*, vol. 15, no. 6, pp. 617–644, Dec. 2009, doi: 10.1007/s10732-008-9080-4.

BIOGRAPHIES OF AUTHORS

Ali Kansou received the Ph.D. degree in computer science from Le Havre University, France, in 2009, with the dissertation hybrid metaheuristics for Arc routing problems. He is currently a lecturer and researcher in the Faculty of Sciences at the Lebanese University. His research interests include combinatorial optimization, arc and vehicle routing problems, hybrid metaheuristics, and applications of artificial intelligence to logistics and transportation. He can be contacted at email: ali.kansou@gmail.com.

Bilal Kanso received the Ph.D. degree in computer science from École Centrale Paris, France, in 2011, with the dissertation modeling and testing of component-based systems. He is currently a lecturer and researcher in the Faculty of Sciences at the Lebanese University. His research interests include formal methods for software engineering, automata theory, specification and testing of temporal properties, as well as metaheuristic algorithms for routing and optimization problems. He can be contacted at email: bilal_kanso@hotmail.com.

Houssein Wehbe is an assistant professor at the American University of Iraq - Baghdad (AUIB), College of Arts and Sciences. He obtained his Ph.D. in Computer Science from the University of Rennes 1, France. Since 2013, he has held assistant professor and lecturer positions at the Lebanese University, the Lebanese International University, and Rafik Hariri University in Lebanon. A certified AWS Solutions Architect and Cisco Instructor, he has also worked as a researcher and developer at Orange Labs and Mines-Télécom/Télécom Bretagne in France. He can be contacted at email: houssein.wehbe@auib.edu.iq.

Haydar Bazzi is an instructor at the Faculty of Information II, Lebanese University. He holds a master in Computer & Telecommunications from the Lebanese University. His research interests include applied optimization, logistics and scheduling, and decision-support systems that combine metaheuristics and mathematical programming with explainable visual analytic. He can be contacted at email: haydar.bazzi@ul.edu.lb.

Ali Mcheik obtained his Ph.D. in 2010 from the University of Toulouse (France), where his research focused on the segmentation of skin OCT images. He is currently a lecturer and researcher at the Lebanese University, Faculty of Sciences. His ongoing work centers on ultrasound imaging and cardiac signal analysis. He can be contacted at email: ali_mch@hotmail.com.