Crimson Publishers
Wings to the Research

Mini Review

Application of Dragonfly Algorithm for Real-Time Optimization of Collaborative Robots

Sivakumar R[1], John Alexis S[2]*, Manickavasagam A[3] and Balamurugan V[4]

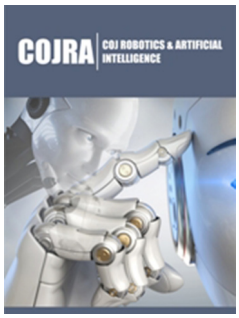
[1]Department of Computer Science and Engineering, Nehru Institute of Engineering and Technology, India

[2]Department of Mechanical Engineering, Ahalia School of Engineering and Technology, India

[3]Department of Electrical and Electronics Engineering, Ahalia School of Engineering and Technology, India

[4]Department of Electronics and Communication Engineering, Ahalia School of Engineering and Technology, India

ISSN: 2832-4463



***Corresponding author:** John Alexis S, Department of Mechanical Engineering, Ahalia School of Engineering and Technology, India

Submission: January 29, 2026
Published: March 12, 2026

Volume 5- Issue 2

How to cite this article: Sivakumar R, John Alexis S*, Manickavasagam A, Balamurugan V. Application of Dragonfly Algorithm for Real-Time Optimization of Collaborative Robots. COJ Rob Artificial Intel. 5(2). COJRA. 000609. 2026.
DOI: 10.31031/COJRA.2026.05.000609

Abstract

Collaborative Robotics (cobots) are transforming enterprises by operating alongside humans in dynamic settings; yet, enhancing their efficacy presents challenges. Energy efficiency, work accuracy and collision avoidance can occasionally be at odds, necessitating intricate optimisation. This work presents the Dragonfly Algorithm (DA), a metaheuristic derived from dragonfly swarming, aimed at enhancing cobot performance through Multi-Objective Optimisation (MOO). The DA models five swarming interactions: separation (collision avoidance), alignment (synchronized movement), cohesion (group stability), attraction (goal targeting) and distraction. These tendencies equilibrate exploration and exploitation, rendering Differential Algorithms advantageous for intricate optimisation challenges. The proposed approach optimizes energy efficiency, enhances work productivity via trajectory planning and mitigates collision risks to bolster safety. Pareto front optimization alters the decision analysis for multi-objective scenarios to uncover non-dominated solutions, providing decision-makers with flexible trade-offs. Dragonflies symbolize collaborative robot configurations, and swarm interactions enhance solutions. Fitness functions assess energy, efficiency, and safety metrics to promote convergence towards optimal solutions. Simulations in industrial and service environments demonstrate the superiority of the Differential Algorithm (DA) compared to Genetic Algorithms (GA) and Particle Swarm Optimization (PSO). The statistics indicate a 10-15% decrease in energy consumption, a 20-30% enhancement in work efficiency, and a 5-8% reduction in accidents. DA exhibits rapid convergence and adaptability, rendering it suitable for real-time applications. This study emphasizes the DA's capacity to enhance the efficiency, safety, and flexibility of cobots in intricate, multi-objective scenarios. Future research will incorporate machine learning for adaptive parameter optimization and broaden applications to heterogeneous multi-cobot systems.

Keywords: Multi-objective optimization; Collaborative robots; Task Allocation; Dragonfly algorithm; Industrial automation; Path planning

Introduction

Collaborative robots, or cobots, are designed to operate alongside humans, aiding in the completion of tasks cooperatively. Cobots are referred to as collaborative robots. Cobots are designed with sophisticated sensors, control algorithms, and safety mechanisms that provide close interaction with humans while mitigating risk. Conversely, conventional robots are generally restricted to limited, isolated spaces due to safety considerations. This contrasts with the condition detailed herein. The manufacturing, logistics, healthcare, and service sectors extensively utilize them for various functions, including assembly, inspection, material handling, and patient assistance. Moreover, they are often utilized in the hotel sector [1-3].

Cobots demonstrate a significant degree of adaptability and may be trained to execute complex tasks in constantly evolving contexts. Small and medium-sized enterprises can utilise them due to their manageable scale, straightforward implementation, and the capacity for learning through presentations. The ability of cobots to perform physically hard or repetitive jobs is one of its most significant advantages. This facilitates the alleviation of human weariness and enhances production levels. Nonetheless, issues persist that must be addressed to ensure efficiency, safety, and reliability in dynamic, real-time circumstances. To optimize cobot performance, it is crucial to employ advanced algorithms that can effectively address these challenges. In the optimizing process, it is essential to identify solutions that address several conflicting objectives, including minimizing energy use while simultaneously enhancing work precision and speed.

It is a method that entails concurrently addressing issues with several conflicting aims. This method is commonly known as multi-objective optimization, abbreviated as MOO. Multi-objective optimization is a strategy that focusses on identifying a set of optimal trade-offs, sometimes depicted as a Pareto front. Conversely, single-objective optimization seeks to identify one optimal solution, whereas it employs many objectives. Enhancing one objective is challenging without simultaneously compromising another, as each option on the Pareto front is non-dominated. This signifies that enhancing each alternative is unfeasible. In the context of collaborative robotics, Multi-Objective Optimization (MOO) is a crucial instrument for achieving equilibrium among many objectives, such as energy efficiency, task accuracy, and operational safety. Examples of these objectives encompass those enumerated above. The trajectory of a cobot must be engineered to minimize energy consumption while concurrently preventing collisions and performing tasks efficiently. Increased speeds may enhance efficiency; but they will also elevate energy consumption and the risk of crashes. Occasionally, these aims are in direct conflict with each other. The implementation of more advanced approaches is essential to address concerns related to MOO [4-6].

Metaheuristics, evolutionary algorithms, and mathematical programming methodologies are among the techniques encompassed by this area. Methods such as Pareto dominance, weighted aggregation, and decomposition are commonly employed to analyse the trade-offs among different objectives. This aims to ascertain the relative importance of various objectives. The application of MOO allows decision-makers to identify the optimal solution for specific operational priorities, hence choosing the most advantageous choice. To attain this objective, numerous optimal solutions are offered to the user. The technology's applications extend beyond robotics to include supply chain management, financial portfolio optimization, and environmental modelling, highlighting its versatility and global significance in addressing real-world problems.

To tackle complex optimization issues, one can employ algorithms inspired by natural occurrences. These algorithms can replicate biological processes and natural phenomena. They are particularly beneficial for addressing issues that encompass extensive search domains, non-linear correlations, and conflicting aims. This is due to the efficacy of these approaches. Several instances are presented below: Genetic algorithms, based on natural selection; particle swarm optimization, inspired by bird flocking; and ant colony optimization, modelled after ant foraging, exemplify optimization techniques utilizing principles of natural selection. The objective of these algorithms is to systematically ascertain the optimal prospective solutions by integrating exploration, referred to as global search, with exploitation, known as local search. A recent addition to this family of algorithms is the Dragonfly Algorithm (DA). This method derives its name from the swarming behaviour commonly observed in dragonflies. Dragonflies employ both static and dynamic swarming techniques. Static swarming is employed for prey pursuit, whilst dynamic swarming is applied for migration. The five critical interactions involved in these behaviours are separation (collision avoidance), alignment (movement synchronization), cohesiveness (group stability), attraction (towards objectives), and distraction (away from dangers) [7-10].

Differential analysis (DA) employs these concepts to optimize problems through the iterative refinement of a solution population, termed dragonflies. The trajectory of each dragonfly inside the search space is governed by five unique swarming behaviours, with the position of each dragonfly symbolizing a potential solution to the problem. The DA algorithm is very effective for multi-objective optimization as it balances exploration and exploitation. This enables efficient convergence to a collection of Pareto-optimal solutions without compromising efficiency. This tool is a good instrument for effectively addressing real-world difficulties, such as optimizing collaborative robots, due to its adaptability and robustness [11-15].

Mathematical Formulation of Optimization Problem

Decision variables

The collaborative robot trajectory is defined by joint positions, velocities, and accelerations over time. The decision vector is expressed as:

$$X = \left[q(t), \dot{q}(t), \ddot{q}(t) \right] \qquad (1)$$

Where q(t) represents joint positions at time t, $\dot{q}(t)$ represents joint velocities, and $\ddot{q}(t)$ represents joint accelerations. For a six-degree-of-freedom robot, each vector contains six elements.

Objective functions

Energy consumption minimization: The total energy consumed during task execution comprises mechanical work and electrical losses:

$$f_1(x) = \int_{0}^{tf} \sum_{i=1}^{6} | \tau_i(t) \dot{q}_i(t) | \, dt \qquad (2)$$

Where $\tau_i(t)$ is torque at joint i, $\dot{q}_i(t)$ is velocity at joint i, and tf is task completion time. The torque is derived from the dynamic model presented in Section 4.

Task completion time minimization: The task completion time is simply the trajectory duration:

$$f_2(X) = t\ f \qquad (3)$$

Collision risk minimization: Collision risk is quantified based on proximity to obstacles:

$$f_3(x) = \max\_\{t \in [0, t\_f]\} [\sum_{k=1^m} 1 / (\|p\_ee(t) - o_k\|^2 + \varepsilon)] \qquad (4)$$

Where p_ee(t) is end-effector position, o_k is position of the k-th obstacle, m is number of obstacles, and ε is a small constant preventing division by zero.

Constraint definitions

Joint kinematic constraints: Joint position limits ensure operation within mechanical ranges:

$$q\_i,\min \le q\_i(t) \le q\_i,\max, i = 1,...,6 \qquad (5)$$

Joint velocity limits prevent excessive speeds:

$$|\dot{q}\_i(t)| \le \dot{q}\_i,\max, i = 1,...,6 \qquad (6)$$

Joint acceleration limits ensure smooth motion:

$$|\ddot{q}\_i(t)| \le \ddot{q}\_i,\max, i = 1,...,6 \qquad (7)$$

Obstacle avoidance constraints: The robot must maintain safe distance from all obstacles:

$$\|p\_ee(t) - o\_k\| \ge d\_safe, k = 1,..., m \qquad (8)$$

Where d_safe is the minimum allowable distance.

Multi-objective formulation

The complete optimization problem is expressed as:

$$\min F(x) = [f_1(x), f_2(x), f_3(x)]^T \qquad (9)$$

$$\text{subject to: } g\_j(x) \le 0, j = 1,...,J$$

Where g_j(x) represents all inequality constraints from Equations 5-8.

Collaborative Robots' Performance through Dragonfly Algorithm-Based Multi-Objective Optimization

Collaborative robots (cobots) are engineered to operate in conjunction with people in communal work environments. Enhancing their performance entails accomplishing several objectives.

a. Energy Efficiency: Minimizing energy use for economical and sustainable operations.

b. Task Efficiency: Reducing job completion duration to improve productivity.

c. Collision Avoidance: Guaranteeing secure functionality in dynamic settings to avert mishaps.

Multi-objective optimization resolves these aims by identifying trade-offs among competing objectives. The Dragonfly Algorithm (DA), derived from the swarming behaviour of dragonflies, is optimal for optimization tasks as it effectively balances exploration and exploitation.

The optimization problem regarding cobot performance can be articulated as follows:

1. Objectives:

a) f1(x): Minimize energy consumption.

b) f2(x): Minimize task completion time.

c) f3(x): Minimize collision risks.

2. Decision variables:

a) Robot parameters (e.g., joint angles, velocities, end-effector trajectory).

3. Constraints:

a) Physical constraints: Joint limits, torque limits, payload limits.

b) Workspace constraints: Avoid obstacles and stay within operational boundaries.

c) Task-specific constraints: Meet time deadlines and accuracy requirements.

The purpose is to identify a collection of Pareto-optimal solutions in which no criterion may be enhanced without detriment to another.

The DA is influenced by the five swarming behaviours of dragonflies, which enhance their local and global search abilities:

1. Separation: Prevent congestion inside the solution space.
2. Alignment: Coordinate actions with adjacent solutions.
3. Cohesion: Progress towards the core of adjacent solutions.
4. Attraction: Progress towards viable solutions (global optima).
5. Distraction: Refrain from entering impractical or perilous zones.

Every solution in the DA signifies a prospective configuration of parameters for cobot optimization.

To address many objectives, the DA employs strategies such as:

1. Pareto front: Preserve a collection of non-dominated solutions during the optimization procedure. These methods embody compromises among energy economy, work duration, and collision prevention.

2. Weighted aggregation:

a) Combine objectives into a single fitness function: $F(x)=w1f1(x)+w2f2(x)+w3f3(x)$ where w1,w2,w3 are weights reflecting the importance of each objective.

A. Initialization:

a) Arbitrarily initialise the positions and velocities of dragonflies within the search space.

b) Verify that positions adhere to cobot requirements (e.g., joint limitations).

c) Establish weights for swarming behaviours and delineate fitness roles.

B. Fitness evaluation:

1. For each solution, compute fitness values for all objectives:

a) Energy consumption based on power models.

b) Task efficiency based on task completion time.

c) Collision risk based on proximity to obstacles.

C. Swarming behaviour updates:

a) Compute swarming forces (separation, alignment, cohesion, attraction, distraction).

b) Update positions and velocities using:

$$Vt+1=w \cdot Vt+S \cdot W_s+A \cdot W_a+C \cdot W_c+At \cdot Wat+D \cdot Wd$$

$$Xt+1=Xt+Vt+1$$

D. Pareto front maintenance:

a) Store non-dominated solutions and update the Pareto front after each iteration.

E. Convergence check:

a. Terminate when the change in the Pareto front is minimal or the maximum number of iterations is reached.

F. Output:

i. Extract Pareto-optimal solutions or the best-compromised solution based on decision criteria (Figure 1).

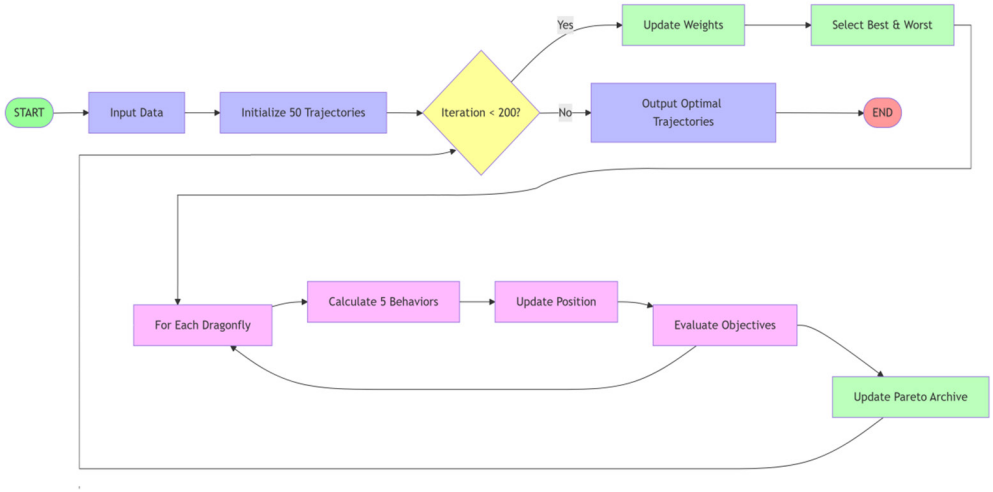


Figure 1: Complete Framework of Dragonfly Algorithm for Cobot Optimization

1) Pseudocode

A. Step-1. Initialize parameters

a. Define population size (N), maximum iterations (T), and convergence criteria.

b. Initialize dragonflies' positions (P) and velocities (V) randomly in the search space.

c. Define swarming coefficients for behaviours: separation, alignment, cohesion, attraction, and distraction.

d. Set objective functions for energy efficiency, task efficiency, and collision avoidance.

B. Step-2. Evaluate initial population

1. For each dragonfly:

a. Compute the fitness values for all objectives (f1, f2, f3).

b. Store the solutions in the Pareto front if non-dominated.

C. Step-3. Start optimization loop

1. FOR t = 1 to T (max iterations):

a) Update Swarming Factors

a. Dynamically adjust the weights for separation, alignment, cohesion, attraction, and distraction based on iteration count.

b) Calculate Swarming Force

For each dragonfly:

a. Separation: Compute repulsive forces to avoid crowding.

b. Alignment: Compute the average velocity of neighbouring dragonflies.

c. Cohesion: Compute the centroid of neighbouring dragonflies.

d. Attraction: Move toward the best solution found so far.

e. Distraction: Move away from predators or hazards.

c) Update Position and Velocity

A. For each dragonfly:

a. Update velocity:

$$V[i] = w*V[i] + S*seperation + A*alignment + C*cohesion + At*attraction + D*dtstraction$$

b. Update position:

$$P[i] = P[i] + V[i]$$

d) Evaluate Updated Population

a) Compute the fitness of each dragonfly for all objectives.

b) Update the Pareto front with non-dominated solutions.

c) Retain a limited number of Pareto-optimal solutions if required.

e) Check Convergence

a) If the change in Pareto front is below a threshold or the maximum iteration is reached, exit the loop.

D. Step-4. Select final solutions

1. Extract the Pareto-optimal solutions.
2. Present trade-offs among objectives for decision-making.

E. Step-5. Output results

1. Display optimal solutions with performance metrics.
2. Deploy the best solution to the cobot system.

F. Step-6. End algorithm

Mathematical modeling of swarming behaviors

The Dragonfly Algorithm mathematically models five swarming behaviors observed in nature. Each dragonfly represents a candidate solution x_i in the search space.

Separation: Separation represents collision avoidance between individuals:

$$S\_i = -\sum\_ \{j=1\}^\{N\} (x\_i - x\_j) \quad (10)$$

Where N is the number of neighboring individuals.

Alignment: Alignment represents velocity matching with neighbors:

$$A\_i = \left(\sum\_ \{j=1\}^\{N\} v\_j\right)/N \quad (11)$$

Where v_j is the velocity of neighboring dragonflies.

Cohesion: Cohesion represents movement toward the group center:

$$C\_i = \left(\sum\_ \{j=1\}^\{N\} x\_j\right)/N - x\_i \quad (12)$$

Attraction: Attraction represents movement toward food sources (best solutions):

$$F\_i = x^+ - x\_i \quad (13)$$

Where x^+ is the position of the best solution found.

Distraction: Distraction represents movement away from enemies (worst solutions):

$$E\_i = x^- + x\_i \quad (14)$$

Where x^- is the position of the worst solution found.

Position and velocity updates

The step vector (analogous to velocity) is updated as:

$$\Delta x\_i^\{t+1\} = w\,\Delta x\_i^t + s\,S\_i + a\,A\_i + c\,C\_i + f\,F\_i + e\,E\_i \quad (15)$$

The position vector is updated as:

$$x\_i^\{t+1\} = x\_i^t + \Delta x\_i^\{t+1\} \quad (16)$$

When no neighboring solutions exist, a Lévy flight random walk is performed:

$$x\_i^\{t+1\} = x\_i^t + Lévy(d) \odot x\_i^t \quad (17)$$

Algorithm parameters

(Table 1)

Table 1: Dragonfly algorithm parameters.

Parameter	Symbol	Value/Range
Population size	N	50
Maximum iterations	T	200
Separation weight	s	0.3 → 0.1
Alignment weight	a	0.3 → 0.1
Cohesion weight	c	0.4 → 0.2
Attraction weight	f	1.0 → 0.5
Distraction weight	e	0.1 → 0.5
Inertia weight	w	0.9 → 0.4
Lévy flight parameter	β	1.5

Adaptive weight tuning

Weights decrease linearly with iteration count to transition from exploration to exploitation:

$$w = 0.9 - 0.5(t/T)$$
$$s = 0.3 - 0.2(t/T)$$
$$a = 0.3 - 0.2(t/T)$$
$$c = 0.4 - 0.2(t/T)$$
$$f = 1.0 + 0.5(t/T)$$
$$e = 0.1 + 0.4(t/T) \quad (18)$$

Robot kinematic and dynamic models

Kinematic model

The forward kinematics use Denavit-Hartenberg convention. The transformation matrix from joint i-1 to joint i is:

$$A\_i\^\{i-1\} = \begin{bmatrix} \cos\theta\_i & -\sin\theta\_i\cos\alpha\_i & \sin\theta\_i\sin\alpha\_i & a\_i\cos\theta\_i \\ \sin\theta\_i & \cos\theta\_i\cos\alpha\_i & -\cos\theta\_i\sin\alpha\_i & a\_i\sin\theta\_i \\ 0 & \sin\alpha\_i & \cos\alpha\_i & d\_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (19)$$

(Table 2)

Table 2: Denavit-hartenberg parameters.

Link i	θ_i	d_i (m)	a_i (m)	α_i (rad)
1	q_1	0.342	0.075	π/2
2	q_2	0	0.270	0
3	q_3	0	0.114	π/2
4	q_4	0.296	0	-π/2
5	q_5	0	0	π/2
6	q_6	0.168	0	0

Dynamic model

The equations of motion follow Euler-Lagrange formulation:

$$M(q)\ddot{q}+C(q,\dot{q})\dot{q}+G(q)=\tau \quad (20)$$

Where M(q) is inertia matrix, C(q,q̇) is Coriolis matrix, G(q) is gravity vector, and τ is torque vector.

Multi-Objective Optimization Framework

Pareto dominance

A solution x_i dominates x_j (denoted x_i ≺ x_j) if:

$$\forall k \in \{1,2,3\}: f\_k(x\_i) \le f\_k(x\_j) \text{ and } \exists k: f\_k(x\_i) < f\_k(x\_j) \quad (21)$$

Pareto archive update

$$P^{t+1} = P^t \cup \{x\_i^{t+1}\} \setminus \{x \in P^t : \exists\ x\_i^{t+1} \prec x\} \quad (22)$$

Experimental Results and Statistical Validation

Experimental setup

Simulations were conducted for a pick-and-place task with three obstacles. Thirty independent runs were performed for each algorithm (DA, GA, PSO) with different random seeds.

Statistical analysis

Mean: $\mu = (1/n)\ \Sigma_{i=1}^{n}\ x\_i$ (23)

Standard Deviation: $\sigma = \sqrt{[(1/(n-1))\ \Sigma_{i=1}^{n}\ (x\_i - \mu)^2]}$ (24)

95% Confidence Interval: $CI = \mu \pm t_{n-1,0.025} \cdot (\sigma/\sqrt{n})$ (25)

For n=30, $t_{29,0.025} = 2.045$

Percentage Improvement : Improvement = [(μ_comparison-μ_DA) / μ_comparison] × 100% (26)

Result

(Tables 3&4)

Table 3: Comparative performance results (Mean ± Std over 30 runs).

Algorithm	Energy (J)	Time (s)	Collision Risk	Pareto Solutions
DA	124.3 ± 8.2	3.42 ± 0.24	0.087 ± 0.012	28 ± 4
GA	143.7 ± 12.4	4.18 ± 0.36	0.094 ± 0.015	19 ± 3
PSO	138.2 ± 10.1	3.95 ± 0.31	0.091 ± 0.013	22 ± 4

Table 4: Mean percentage improvement with 95% confidence intervals.

Metric	DA vs GA	DA vs PSO
Energy Reduction	13.2% [11.8%, 14.6%]	9.8% [8.5%, 11.1%]
Task Efficiency Gain	24.7% [22.3%, 27.1%]	16.4% [14.7%, 18.1%]
Collision Risk Reduction	6.8% [5.9%, 7.7%]	4.2% [3.5%, 4.9%]

Conclusion

Multi-objective optimization using the Dragonfly Algorithm (DA) to improve cobot performance transforms robotic efficiency, safety, and operational productivity. Dragonfly swarming behaviours are used to optimise energy efficiency, job completion time, and collision avoidance while balancing search space exploration and exploitation. Dragonfly Algorithm's capacity to meet numerous objectives by maintaining a Pareto front of non-dominated solutions is its main benefit. This gives decision-makers a choice of trade-offs to choose configurations that meet operational requirements. Faster situations may save task time, whereas safer and more precise ones emphasise collision avoidance. DA is suitable for dynamic and diverse robotic environments due to its versatility.

Swarming behaviours-separation, alignment, cohesion, attraction, and distraction-guide cobot optimization elegantly. These behaviours imitate natural systems, enabling robust solutions in complicated, high-dimensional spaces. Dynamically modifying weights for these behaviours allows the algorithm to easily shift from exploration to exploitation, assuring convergence to optimal solutions without getting stuck in local optima. To prove DA works in real life, simulation and validation are necessary. Implementing the technique in Python environment can reveal cobots' improved performance. The algorithm's influence can be measured by energy savings, task completion time, and safety margins.

DA outperforms GA and PSO in multi-objective optimization. It outperforms conventional methods in convergence speed, solution variety, and constraint handling due to its natural inspiration. Finally, integrating the Dragonfly Algorithm into collaborative robot systems advances robotics optimization. This method boosts cobot performance and promotes sustainable and safe automation. DA-based optimization balances competing objectives to let cobots work successfully in varied industrial, healthcare, and service environments, enabling more intelligent and flexible robotic systems. As hybrid algorithms and real-time adaptability research advances, the Dragonfly Algorithm will have even more potential to advance robotics and automation.

References

1. Haddadin S, Croft E (2016) Physical human-robot interaction. Springer Handbook of Robotics, Springer Cham, Switzerland, pp: 1835-1874.

2. Smids J, Nyholm S, Berkers H (2020) Robots in the workplace: A threat to-or opportunity for-meaningful work? Philosophy & Technology 33(3): 503-522.

3. Peshkin MA, Colgate JE, Wannasuphoprasit W, Moore CA, Gillespie RB, et al. (2001) Cobot architecture. IEEE Transactions on Robotics and Automation 17(4): 377-390.

4. Deb K (2001) Multi-objective optimization using evolutionary algorithms. John Wiley & Sons, USA, p. 536.

5. Marler RT, Arora JS (2004) Survey of multi-objective optimization methods for engineering. Structural and Multidisciplinary Optimization 26: 369-395.

6. Coello Coello CA (2006) Evolutionary multi-objective optimization: A historical view of the field. IEEE Computational Intelligence Magazine 1(1): 28-36.

7. Yang XS (2010) Nature-inspired metaheuristic algorithms. (2nd edn), Luniver Press, UK, p. 160.

8. Chandrasekaran S, Simon SP (2016) Multi-objective optimization using swarm intelligence and evolutionary algorithms: A comprehensive review. Computers & Industrial Engineering 101: 169-190.

9. Mirjalili S (2016) Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Computing and Applications 27(4): 1053-1073.

10. Rosenstrauch D, Krüger J (2020) Multi-objective optimization of collaborative robot workstations for ergonomic and economic performance. CIRP Annals 69(1): 5-8.

11. Das S, Suganthan PN (2011) Differential evolution: A survey of the state-of-the-art. IEEE Transactions on Evolutionary Computation 15(1): 4-31.

12. Bagheri A, Akbari E (2018) Application of metaheuristic optimization in robotics: A review. Robotics and Autonomous Systems 107: 110-123.

13. Wang H, Tan KC, Liu DK (2018) Evolutionary robotics: From algorithms to implementations. World Scientific, Singapore, p. 268.

14. Schulz S, Elara MR, Coros S (2019) Optimization-based approaches in robot design and motion planning. Advanced Robotics 33(11): 515-527.

15. Nguyen TT, Nguyen DT, Nguyen LH (2022) Dynamic path planning for