

Smart Robot Control via Novel Computational Intelligence Methods for Ambient Assisted Living

Bo Xing

Abstract In recent years, we have witnessed a rapid surge in ambient assisted living (AAL) technologies due to a rapidly aging society. The aging population, the increasing cost of formal health care, the caregiver burden, and the importance that the individuals place on living independently, all motivate development of innovative-assisted living technologies for safe and independent aging. Among all areas, the development of assistive robots with potential application to health and elderly care is drawing a lot attention from both industry and academia. Several such prototypes (e.g., wearable assisted-walking device designed by Honda, and personal transport assistance robot designed by Toyota) have been made. These robots are inevitably supported by various algorithms and computational techniques. In this work, we provide an overview of applying emerging CI (i.e., non-conventional CI) approaches to various smart robot control scenarios which, from the author's viewpoint, have a great influence on various ambient assisted living (AAL) robot related activities, such as location identification, manipulation, communication, vision, learning, and docking capabilities. The innovative CI methods covered in this chapter include bacteria foraging optimization (BFO), bees algorithm (BA), glowworm swarm optimization (GSO), grey wolf optimizer (GWO), electromagnetism-like mechanism (EM), intelligent water drops (IWD), and gravitational search algorithm (GSA). The findings of this work can provide a good source for someone who is interested in the research filed of AAL robot.

Keywords Ambient assisted living (AAL) • Robot control • Computational intelligence (CI) • Bacteria foraging optimization (BFO) • Bees algorithm (BA) • Glowworm swarm optimization (GSO) • Electromagnetism-like mechanism (EM) • Intelligent water drops (IWD) • Gravitational search algorithm (GSA) • Grey wolf optimizer (GWO)

B. Xing (✉)

Computational Intelligence, Robotics, and Cybernetics for Leveraging E-future (CIRCLE),
Department of Computer Science, School of Mathematical and Computer Sciences,
Faculty of Science and Agriculture, University of Limpopo,
Private Bag X1106, 0727 Sovenga, Limpopo, South Africa
e-mail: bxing2009@gmail.com

1 Introduction

Technology and automation have made a great number of labour-saving robots become possible. For example, numerous monotonous and dangerous tasks such as sheet metal painting and oil pipelines cleaning used to be completed by humans have been substituted by robots. More recently, the implementation of ambient assisted living (AAL) robots that are capable of assisting human beings, in particular in the scenario of offering assistance to the disabled and the elderly, has become more and more popular (e.g., MOVAID system [1, 2]). However, there are times that these automated devices poorly perform, especially in the situation of when they really needs to interact with another human being [3]. At present, different computational intelligence (CI) approaches such as neural networks [4], reinforcement learning [5–7], and fuzzy logic [8] are helping us to enable a higher level of robot intelligence. Nevertheless, in spite of these algorithms’ tremendous success, none of them performs predominant over the rest. To overcome this issue, researchers come up two solutions: one is to use the hybridized version of these algorithms in which the strengths of each algorithm is enhanced while the weaknesses are largely compensated or eliminated (e.g., [9–14]); the other one is to develop new intelligent algorithms which can perform more powerful than their predecessors. Although the first trend is currently well-documented, the second trend is being less mentioned in the literature. The focus of this study is thus placed on the latter trend that is, exploring the novel CI methods and pinpointing their applications in the field of robot control with a particular emphasis, if possible, on AAL robot relevant studies.

With this in mind, the remainder of this chapter is organized as follows: First, Sect. 2 briefly introduces the relevant terms used in this study; Second, the innovative CI algorithms (i.e., biology- and physics-based CI) and their selected applications in smart robot control are detailed in Sects. 3 and 4, respectively; an experimental study that implementing an innovative CI algorithm (i.e., grey wolf optimizer) for the heterogeneous robot swarm control has been described in Sect. 5, which is followed by future work highlighted in Sect. 6; Finally, Sect. 7 draws the conclusion of this study.

2 Background

2.1 What Is Ambient Assisted Living (AAL) Robot?

The main purpose of introducing AAL robots is to assist the disabled and elderly people at home [15–17]. In general, they can be categorized into three categories: robots assisting with daily living activities (such as feeding and dressing), robots

assisting with instrumental activities of daily living (such as housekeeping and preparing food), and robots assisting with enhanced activities of daily living (such as communication and engaging in hobbies) [16]. Nowadays, a number of assistive robots are being deployed. For example, “Care-O-bot” [18–20], a robot developed by Fraunhofer IPA, is able to fetch and carry objects, communicate with older people, and supply emergency support; “RIBA” [21, 22], a robot developed by RIKEN-TRI Collaboration Center, can help patient transfer; “uBot5” [23], another robot from the University of Massachusetts Amherst, is capable of achieving multiple postures for the purpose assisting elderly in compensating for impaired upper extremity function; “PerMMA” [24–26], a research outcome from the Carnegie Mellon University and the University of Pittsburgh, can assist persons with disabilities; “PaPeRo” [27–29], another case developed by NEC, is used to communicate; and “Emiew” [30–32] developed by Hitachi, can interact with human beings; and “Hospi-R” [33, 34], an autonomous delivery robot developed by Matsushita, can even perform complex service tasks.

2.2 *What Is Robot Control?*

As we know, designing assistive living robots is a multi-dimensional, interdisciplinary challenge, which including design challenges, pervasive computing challenges, social challenges, safety challenges, etc. Among these challenges, the control of an autonomous mobile robot is a complex, usually non-linear and partly unpredictable process. The most distinguished difference, between robots and other technologies, lies in that robots’ capability to combine automation with action, and in the meantime a considerable amount of mobility. Static tasks only form part of their job can-do list, robots can also do lots of dynamic jobs such as exploration, walking, and lifting. Robots are playing an increasingly important role in both scientific and our daily life. The ultimate goal of the robot control is thus to let robots seamless integrated into a human environment, or in other words, to become more and more humanoid.

2.3 *What Is Novel Computational Intelligence?*

Computational intelligence (CI) is a fairly new research field, which is still in a process of evolution. At a more general level, CI comprises a set of computing systems with the ability to learn and deal with new events/situations, such that the systems are perceived to have one or more attributes of reason and intelligence [35]. According to their popularity, the CI methods can be roughly classified into two categories, namely, traditional CI and novel CI. Typically, traditional CI approaches

include such as simulated annealing, genetic algorithm, particle swarm optimization, ant colony optimization, Tabu search, and artificial neural network. In terms of novel CI approaches [36], this overview covers the following candidates: bacterial foraging optimization (BFO), bees algorithm (BA), glowworm swarm optimization (GSO), electromagnetism-like mechanism (EM), and intelligent water drops (IWD).

3 Biology-Based CI

In this section, we detail a group of biology-based algorithms which get their inspiration from different animals. The representative applications of these algorithms in dealing with various robot control problems are also highlighted.

3.1 *Bacteria Foraging Optimization (BFO) Algorithm*

Bacterial foraging optimization (BFO) algorithm was originally proposed in Passino [37] where the foraging strategy of *E. Coli* bacteria has been simulated. Typically, the BFO consists of four main mechanisms: chemotaxis, swarming, reproduction, and elimination-dispersal event. The main steps of BFA are outlined as below [37]:

- Chemotaxis: The movement of *E. coli* bacteria can be performed through two different ways: swimming which means the movement in the same direction, and tumbling refers the movement in a random direction. The movement of the i th bacterium after one step is given by Eqs. (1) and (2), respectively [37]:

$$\theta^i(j+1, k, l) = \theta^i(j, k, l) + C(i)\phi(j). \quad (1)$$

$$\begin{cases} \theta^i(j+1, k, l) > \theta^i(j, k, l), & \text{swimming in which } \phi(j) = \phi(j-1) \\ \theta^i(j+1, k, l) < \theta^i(j, k, l), & \text{tumbling in which } \phi(j) \in [0, 2\pi] \end{cases}. \quad (2)$$

where $\theta^i(j, k, l)$ denotes the location of i th bacterium at j th chemotactic, k th reproductive and l th elimination and dispersal step, $C(i)$ is the length of unit walk, and $\phi(j)$ is the direction angle of the j th step.

- Swarming: Under the stresses circumstances, the bacteria release attractants to signal bacteria to swarm together, while they also release a repellant to signal others to be at a minimum distance from it. The cell to cell signalling can be represented by Eq. (3) [37]:

$$\begin{aligned}
J_{cc}(\theta, P(j, k, l)) &= \sum_{i=1}^S J_{cc}^i(\theta, \theta^i(j, k, l)) \\
&= \sum_{i=1}^S \left[-d_{attract} \exp \left(-\omega_{attract} \sum_{m=1}^P (\theta_m - \theta_m^i)^2 \right) \right] \\
&\quad + \sum_{i=1}^S \left[h_{repellant} \exp \left(-\omega_{repellant} \sum_{m=1}^P (\theta_m - \theta_m^i)^2 \right) \right]. \quad (3)
\end{aligned}$$

where $J_{cc}(\theta, P(j, k, l))$ is the objective function value to be added to the actual objective function, S is the total number of bacteria, P is the number of variables to be optimized which are present in each bacterium, $\theta = [\theta_1, \theta_2, \dots, \theta_P]^T$ denotes a point in the P -dimensional search domain, $d_{attract}$ is the depth of the attractant released by the cell, $\omega_{attract}$ is a measure of the width of the attractant signal, $h_{repellant}$ is the height of the repellant effect (i.e., $h_{repellant} = d_{attract}$), and $\omega_{repellant}$ is the measure of the width of the repellant.

- **Reproduction:** After N_c chemotaxis steps, the reproduction step should be performed. The fitness value of the bacteria is stored in an ascending order. The working principle is the least health bacteria eventually die and the remaining bacteria (i.e., healthiest bacteria) will be divided into two identical ones and placed at the same location.
- **Dispersion and elimination:** For the purpose to avoid local optima, dispersion and elimination process is performed after a certain number of reproduction steps. According to a present probability (p_{ed}), a bacterium is chosen to be dispersed and moved to another position within the environment.

Taking into account the key phases described above, the steps of implementing BFO can be summarized as follows [37–40]:

- Step 1: Defining the optimization problem, and initializing the optimization parameters.
- Step 2: Iterative algorithm for optimization. In this step the bacterial population, chemotaxis loop ($j = j + 1$), reproduction loop ($k = k + 1$), and elimination and dispersal operations loop ($l = l + 1$) are performed.
- Step 3: If $j < N_c$, go to the chemotaxis process.
- Step 4: Reproduction.
- Step 5: If $k < N_{re}$, go to reproduction process.
- Step 6: Elimination-dispersal.
- Step 7: If $l < N_{ed}$, then go to elimination-dispersal process; otherwise end.

Smart Robot Control using BFO. Based on the basic BFO, an improved version was proposed in [41] to deal with robot navigation problem. Two simulation scenarios, namely, (i) fixed obstacle and target; and (ii) randomly moving obstacles and fixed target, were considered by the authors of [41]. 10 standard deviation values of path lengths were assigned to each testing scenarios where the

obtained results are classified into three groups, i.e., best, worst, and average. By comparison the experiment results, both BFO algorithms (improved and basic version) outperform the traditional particle swarm optimization algorithm. At the end of their study, the authors claimed that BFO is very powerful in fulfilling the robot path navigation task.

3.2 *Bees Algorithm (BA)*

Inspired by natural foraging behavior of honey bees, the bees algorithm (BA) was proposed by the authors of [42]. The basic working principles of BA are listed as follows:

- Step 1: Initializing population with random solutions. At this stage, the BA requires several parameters to be set such as number of scout bees (n), number of sites selected out of n visited sites (m), number of the best sites out of m selected sites (e), number of bees recruited for the best e sites (nep), number of bees recruited for the other ($m - e$) selected sites (nsp), and initial size of patches (ngh) which includes the site, its neighborhood, and the stopping criterion.
- Step 2: Assessing the fitness of the population.
- Step 3: Forming new population while stopping criterion is not met.
- Step 4: Selecting sites for neighborhood search. The bees with the highest fitness values are chosen as “selected bees” at Step 4, and accordingly, the sites visited by them are chosen for neighborhood search.
- Step 5: Recruiting bees for selected sites (more bees for best s sites), and evaluating the fitness value. In Step 5, the BA conducts search in the neighborhood of the selected sites. More bees will be assigned to search around the best e sites. The bees can either be selected directed based on the value of fitness associated with the sites they are visiting, or the fitness values will be used to determine the probability of the bees being selected. Exploration of the surroundings of the best e sites represents more suitable solutions can be made available through recruiting more bees to follow them than the other selected bees. This differential recruitment mechanism, along with scouting strategy, is the core operation of the BA.
- Step 6: Choosing the fittest bee from each patch. The authors of [42] introduced a constraint at this stage that is for each patch, only the bees with the highest fitness value can be selected to form the next bee population. The purpose of introducing such restriction is to reduce the number of points that are going to be explored.
- Step 7: Assigning the remaining bees to do random search, and evaluating their fitness. In the bee population, the remaining bees are assigned randomly around the search space looking for new possible solution candidates.

- Step 8: Terminating the loop when stopping criterion is met. All aforementioned steps will be executed repeatedly until a stopping criterion is met. At the end of each iteration, the population of bee colony consists of two parts: representatives from each selected patch, and other scout bees performing random searches.

Smart Robot Control using BA. Mobile robots for public service have always been a great research interest both for academic and industry. During the past few years, the study of robot swarm and its related topic has become a hot area. In [43], the authors utilized BA to study a group of reconfigurable mobile robots which are designed to provide daily service in hospital environments for different kinds of tasks such as guidance, cleaning, delivery, and monitoring. The fulfilment of each job requires an associated functional module that can be installed onto various robot platforms via a standard connection interface. Since the classic BA focuses mainly on single-objective functional optimization problems, a variant called binary BA (BBA for short) was proposed by the authors of [43] to deal with the multi-objective multi-constraint combinatorial optimization task. In BBA, a bee is describe as two binary matrixes $\mathbf{MR}$ and $\mathbf{RH}$, standing for how to assign the M tasks to the R robots and the R robots to the H homes, respectively. The size of $\mathbf{MR}$ is $M \times R$ in which its R columns represent the R robots, while the M missions is represented by the M rows.

The authors evaluated the BBA with an example problem (20 missions, 8 robots, and 4 homes) with a size of $8^{20} \times 4^8 = 2^{76}$ combinations. At first, 12 stochastic solutions are obtained by scout bees through global search in which six elite bees survive after the non-dominated selection. The final experiments demonstrated that BBA is a suitable candidate tool in treating workload balancing issue among a team of swarm robots.

3.3 Glowworm Swarm Optimization (GSO) Algorithm

The glowworm swarm optimization (GSO) algorithm was proposed by the authors of [44]. This algorithm gets its inspiration from the behavior from glowworms and it shares some common points with other population-based algorithms such as ant colony optimization and particle swarm optimization. The agents in GSO are a group of glowworms that carry a luminescent quantity call luciferin. Normally, GSO starts by randomly placing n glowworms in the search space so that they are well distributed. Initially, all glowworms carry an equal quantity of luciferin l_0 . Typically, each iteration of GSO algorithm consists of three rules, namely, luciferin-update rule, movement rule, and transition rule. The details are listed as below.

- Luciferin-update phase: It is the process by which the luciferin quantities are modified. The quantities value can either increase, as glowworms deposit

luciferin on the current position, or decrease, due to luciferin decay. The luciferin update rule is given as follows [44]:

$$l_i(t+1) = (1 - \rho) \cdot l_i(t) + \gamma \cdot J \cdot [x_i \cdot (t+1)]. \quad (4)$$

where $l_i(t)$ denotes the luciferin level associated with the glowworm i at time t , ρ is the luciferin decay constant ($0 < \rho < 1$), γ is the luciferin enhancement constant, and $J(x_i(t))$ stands for the value of the objective function at glowworm i 's location at time t .

- **Movement phase:** During this phase, glowworm i chooses the next position j to move to using a bias (i.e., probabilistic decision rule) toward good-quality individual which has higher luciferin value than its own. In addition, based on their relative luciferin levels and availability of local information, the swarm of glowworms can be partitioned into subgroups that converge on multiple optima of a given multimodal function. The probability of moving toward a neighbour is given as follows [44]:

$$p_{ij}(t) = \frac{l_j(t) - l_i(t)}{\sum_{k \in N_i(t)} [l_k(t) - l_i(t)]}. \quad (5)$$

where $j \in N_i(t)$, $N_i(t) = \{j : d_{ij}(t) < r_d^i(t); l_i(t) < l_{js}(t)\}$ is the set of neighbours of glowworm i at time t , $d_{ij}(t)$ denotes the Euclidean distance between glowworms i and j at time t , and $r_d^i(t)$ stands for the variable neighbourhood range associated with glowworm i at time t .

Based on the above equation, the discrete-time model of the glowworm movements can be stated as follows [44]:

$$x_i(t+1) = x_i(t) + s \left[\frac{x_j(t) - x_i(t)}{\|x_j(t) - x_i(t)\|} \right]. \quad (6)$$

where $x_i(t) \in \mathbf{R}^m$ is the location of glowworm i at time t in the m -dimensional real space, $\|\cdot\|$ denotes the Euclidean norm operator, and s (>0) is the step size.

- **Neighbourhood range update rule:** In addition to the luciferin value update rule that is illustrated in the movement phase, in GSO the glowworms use a radial range (i.e., $(0 < r_d^i \leq r_s)$) update rule to explore an adaptive neighbourhood (i.e., to detect the presence of multiple peaks in a multimodal function landscape). Let r_0 be the initial neighbourhood range of each glowworm (i.e., $r_d^i(0) = r_0 \forall i$), then the updating rule is given as follows [44]:

$$r_d^i(t+1) = \min\{r_s, \max\{0, r_d^i(t) + \beta(n_t - |N_i(t)|)\}\}. \quad (7)$$

where β is a constant parameter, and $n_t \in N$ is a parameter used to control the number of neighbours.

Smart Robot Control using GSO. In [45], the authors built a team of four wheeled-mobile robot named Kinbots, and employ GSO algorithm to fulfill the collective robot control. In the work, they equipped their Kinbots with infrared sensor-based interaction modules which can offer a hardware capability to perform luciferin emission or detection and a behavior of leader-following. In a subsequent study [46], the same authors conducted a preliminary experiment to demonstrate the performance of GSO in robot swarm control. A set of three glowworms (i.e., Kinbots) *A*, *B* and *C* are initially located at the corners of an equilateral triangle (side is 50 cm). Kinbots *A* and *B* remained stationary during the study, and emitted a luciferin value of 128 and 60, respectively, while a luciferin value of 40 was glowed by the Kinbot *C*. At each iteration, the sweep platform performs three tasks: First, homes by turning clock-wise until it make a proper angle with the heading direction of the Kinbots; Second, does a 180° scanning to acquire intensity samples and localize the neighbors; and Third, aligns along the line-of-sight of each neighbor to receive the luciferin value emitted by itself. The sensing phase of the first cycle is completed at 10 s. For simplicity, the authors of [46] introduced a maximum-neighbor selection rule that is a Kinbot chooses to move toward a neighbor which emits maximum luciferin. The simulation results demonstrated the suitability of applying GSO in dealing with the problem of multiple source localization encountered in the domain of robot control.

4 Physics-Based CI

This section introduces a physics-based algorithm, named electromagnetism-like mechanism (EM), which gets its inspiration from electromagnetism theory. The representative application of EM in addressing the issue of smart robot control is also highlighted.

4.1 *Electromagnetism-Like Mechanism (EM) Algorithm*

Similar to that in the elementary electromagnetism, the authors of [47] regarded each sample point as a charged particle that is released to a space. In the proposed electromagnetism-like mechanism (EM) approach, the objective function value is associated with the charge of each point which determines the magnitude of attraction/repulsion of the point over the sample population. In other words, the higher the magnitude of attraction, the better the objective function value. Once we get the value of these charges, we can use them to look for a direction, usually obtained through the evaluation of a combination force that exerts on the point via other points, where each point can move toward in the subsequent iterations. Like the electromagnetic forces, by adding vectorially the forces from each of the other points, we can obtain the required force. Typically, the EM algorithm consists of

four phases, namely, initialization, local search, total force calculation, and the movement. For the rest of this section, we will explain them in detail.

- Initialization: This procedure is used to sample m points randomly from the feasible domain, which is an n -dimensional hyper-cube. Each coordinate of a point is assumed to be uniformly distributed between the corresponding upper and lower bound. After a point is selected from the space, the objective function value for that point is calculated using the function pointer $f(x)$. The initialization phase ends after identifying m points, and the point that carries the best function value is stored in x^{best} .
- Local search: This phase is used to collect the local information for a point x^i . First, the maximum feasible step length is computed according to the parameter δ . Second, for a given i , the improvement in x^i is searched coordinate by coordinate. For a given coordinate, the point x^i is assigned to a tentative point y for storing the initial information. Third, a random number is chosen as the step length and the point y moves along that direction. If the point y finds a better point within local search iterations, the point x^i is replaced by y and the neighborhood search for points i terminates. Finally, updating the current best point.
- Total force calculation: In EM, according to the superposition principle of electromagnetism theory, point i 's power of attraction/repulsion is determined by the charge of each point $i(q^i)$ via Eq. (8) [47].

$$q^i = \exp \left\{ -n \frac{f(x^i) - f(x^{best})}{\sum_{k=1}^m [f(x^k) - f(x^{best})]} \right\}, \forall i. \quad (8)$$

Unlike the electrical charges, no signs are attached to the charge of an individual point in Eq. (8). Instead, the authors of [47] decided the direction of a particular force between two points based on the comparison of their objective function values. Accordingly, the total force F^i exerted on the point i can be calculated via Eq. (9) [47].

$$F^i = \sum_{j \neq i}^m \left\{ \begin{array}{ll} (x^j - x^i) \frac{q^i \times q^j}{\|x^j - x^i\|^2} & \text{if } f(x^j) < f(x^i) \\ (x^i - x^j) \frac{q^i \times q^j}{\|x^j - x^i\|^2} & \text{if } f(x^j) \geq f(x^i) \end{array} \right\}, \forall i. \quad (9)$$

- Movement: After assessing the total force vector F^i , the point i can move in the direction of the force by a random step length as shown in Eq. (10).

$$x^i = x^i + \lambda \frac{F^i}{\|F^i\|} (RNG), \quad \text{for } i = 1, 2, \dots, m. \quad (10)$$

where λ denotes the random step length and is assumed to be uniformly distributed within $[0, 1]$, and RNG is a vector whose components represent the

allowed feasible movement toward the upper bound (u^k) or the lower bound (l^k), respectively, for the corresponding dimension.

Smart Robot Control using EM. Autonomous mobile robots have many possible and promising applications in routine or dangerous tasks such as inter-planetary and underwater exploration, dust suction, and mail delivery. In the area of assistive living robots research, one of the essential problems is the path tracking problem which consists of designing control techniques for the purpose of assuring that the robot can follow a predetermined path. Due to the multivariable and nonlinear nature of this problem, the control strategies design for mobile is a very complex task. In order to deal with this issue, in [48], the authors employed EM to minimize the mobile robot controller's cost function in real time manner. In mobile robot control, the kinematic model is widely used. Normally, in the 2-dimentional Cartesian space, the pose of the robot can be represented through Eq. (11) [48].

$$q = (x, y, \theta)^T. \quad (11)$$

where $(x, y)^T$ is the coordinate of O_r in the reference coordinate system, and θ is the heading direction.

In an ideal condition, the wheels of a mobile robot are assumed non-slidable, and thus, the kinematics model can be expressed through Eqs. (12), (13), and (14) [48].

$$q'(t) = S(q)V(t). \quad (12)$$

$$S(q) = \begin{bmatrix} \cos \theta(t) & -d \sin \theta(t) \\ \sin \theta(t) & d \cos \theta(t) \\ 0 & 1 \end{bmatrix}. \quad (13)$$

$$V(q) = \begin{bmatrix} v(t) \\ w(t) \end{bmatrix}. \quad (14)$$

where $v(t)$ and $w(t)$ denotes the agential and angular velocities of mobile robot, respectively.

The goal of their study is to guarantee that the vehicle follows a pre-determined referent path without the displacement error. Nevertheless, since the destination coordinates vary with time, the authors thus placed a reference virtual robot with the same mathematical shown in Eq. (15) [48] mode on the track.

$$\begin{cases} x'_{ref}(t) = v_{ref}(t) \cos[\theta_{ref}(t)] \\ y'_{ref}(t) = v_{ref}(t) \sin[\theta_{ref}(t)] \\ w'_{ref}(t) = \alpha \theta_{ref}(t)/dt \end{cases} \quad (15)$$

After linearizing Eq. (15) around an operating point and subtracting Eq. (13) from the obtained values, the proposed kinematic model of the error is stated as in Eq. (16) [48].

$$\tilde{x}'(t) = A(t) \cdot \tilde{x}'(t) + B(t) \cdot \tilde{U}(t). \quad (16)$$

where $\tilde{x} = \hat{x} - \hat{x}_{ref}$ stands for the error to the reference robot, $\tilde{U}(t) = U(t) - U_{ref}(t)$ represents the control input error, matrices $A(t)$ and $B(t)$ are the Jacobian of (13) in relation to $\tilde{x}(t)$ and $U(t)$, respectively. By utilizing the Euler's method and considering the distribution, a time-variant discrete linear model can be expressed in Eq. (17) [48].

$$\tilde{x}'(k+1) = A(k) \cdot \tilde{x}'(k) + \tilde{B}(k) \cdot \tilde{U}(k). \quad (17)$$

where $\tilde{B}(k)$ is $B(k)$ when adding disturbance $\bar{\tau}_{ed}$ from dynamic system to this system.

Built on aforementioned analysis, the path tracking problem considered in [48] is interpreted in Eq. (18) [48].

$$\bar{M}(q)V'(t) + \bar{V}_m(q, q')V(t) + \bar{F}(V(t)) + \bar{\tau}_d = \bar{B}_\tau. \quad (18)$$

The authors made the comparisons between two algorithm, namely, EM and reference algorithm. From the simulation results, it is observed that the linear and angular velocities of the target mobile robot optimized by EM method have a faster convergence speed. The study of [48] successfully demonstrated that the EM approach present a good performance in minimizing the cost function. The major advantage of employing EM algorithm was, concluded at the end of [48], the ability to provide an effective and simple predictive control strategy.

4.2 Intelligent Water Drops (IWD) Algorithm

The inspiration of intelligent water drops (IWD) algorithm comes from the water drops that flow into rivers, lakes, and seas. The core concept is that gravitational form of the earth drags the water drops in a river to flow towards their final destination. The author of [49] invented this algorithm in 2007. The basic working principles of the IWD algorithm are explained as follows [50].

- In IWD algorithm, each intelligent water drop is assumed to carry a certain amount of the soil, $soil(IWD)$, and to flow at a current velocity, $velocity(IWD)$. Suppose that an IWD wants to move to the node j from its current node i . The velocity of an IWD is updated through Eq. (19) [50] which is based on the amount of the soil on the arc between these two nodes, represented by $soil(i, j)$.

$$vel^{IWD}(t+1) = vel^{IWD}(t) + \begin{cases} \frac{a_v}{b_v + c_v \times soil^\beta(i,j)} & \text{if } soil^\beta(i,j) \neq \frac{b_v}{c_v} \\ 0 & \text{otherwise} \end{cases} \quad (19)$$

where $vel^{IWD}(t+1)$ stands for the updated velocity of an IWD at the node j , while a_v , b_v , and c_v are constant velocity parameters which are adjustable according to focal problems.

- Meanwhile, the time required for an IWD flowing with the velocity of to move from its current node i to the next j , denoted by $time(i,j; vel^{IWD}(t+1))$, can be calculated via Eq. (20) [50].

$$time(i,j; vel^{IWD}) = \frac{f(i,j)}{vel^{IWD}}. \quad (20)$$

- In IWD algorithm, a fast water drop removes more soil from the path it flows on than a slower water drop which is similar to what happens in nature: fast rivers always make their beds deeper than the slower ones. Therefore after an IWD travels from node vel^{IWD} to node vel^{IWD} , the soil, vel^{IWD} , on the path between these two nodes is computed through Eq. (21) [50].

$$soil(i,j) = \rho_0 \times soil(i,j) - \rho_n \times \Delta soil(i,j). \quad (21)$$

where ρ_0 and ρ_n are positive numbers fall within the interval of $[0, 1]$. Another important strategy that every IWD must have is how to select its next node. It is assumed in the algorithm, an IWD prefers to take a path which has less amount of soil than the other paths. Let node i be the current position of an IWD, then the probability $p_j^{IWD}(j)$ of choosing the path from node i to node j is calculated using Eq. (22) [50].

$$p_j^{IWD}(j) = \frac{f(soil(i,j))}{\sum_{k \notin vc(IWD)} f(soil(i,k))}. \quad (22)$$

where the inverse of the soil between nodes i and j is computed by through Eq. (23) [50].

$$f(soil(i,j)) = \frac{1}{\varepsilon_s + g(soil(i,j))}. \quad (23)$$

where the parameter of ε_s is a small positive constant number (usually $\varepsilon_s = 0.01$) from preventing a potential division by zero, and $g(soil(i,j))$ is normally used to shift the $soil(i,j)$ on the path connecting nodes i and j towards positive values. The mathematical expression of $g(soil(i,j))$ can be found in Eq. (24) [50].

$$g(soil(i,j)) = \begin{cases} soil(i,j) & \text{if } \min_{l \notin vc(IWD)} (soil(i,l)) \geq 0 \\ soil(i,j) - \min_{l \notin vc(IWD)} (soil(i,l)) & \text{otherwise} \end{cases}. \quad (24)$$

- In the proposed algorithm, every created IWD is designed to move from its initial node to the next ones until it finds a solution. At the end of each iteration, the best solution T^{IB} found by the IWDs within such iteration is obtained through Eq. (25) [50].

$$T^{IB} = \arg \max_{\forall T^{IWD}} q(T^{IWD}). \quad (25)$$

- It means that the iteration-best solution T^{IB} is the dominant solution over all other solutions T^{IWD} . At the end of each iteration of the IWD algorithm, the current iteration-best solution T^{IB} is used to update the total best solution T^{TB} as shown in Eq. (26) [50]:

$$T^{TB} = \begin{cases} T^{TB} & \text{if } q(T^{TB}) \geq q(T^{IB}) \\ T^{IB} & \text{otherwise} \end{cases}. \quad (26)$$

Smart Robot Control using IWD. With the recent technological advancement, the development of unmanned vehicular systems, in particular the unmanned combat aerial vehicle (UCAV), have been proved to be beneficial in both military and civilian applications. Nevertheless, the complete benefits of such unmanned systems can only be fulfilled and utilized when their operations could achieve an autonomous level. One of the key requirements for realizing such autonomy is the ability of detecting internal and external changes, and reacting to them in a safe and efficient manner, especially without the intervention from their human operators. Under such circumstance, the trajectory planning becomes a nontrivial task. Typically, the goal of trajectory planning is to generate a space path between an initial location and the desired destination that has an optimal or near-optimal performance under different constraint conditions. In [51], the authors made an attempt to solve this imperative task by utilizing IWD algorithm. According to [51], the focal problem can be briefly described as follows:

Suppose the flight task for an UCAV is from node T^{IB} to node T^{IB} . The mathematical expression of the path connecting these two nodes is shown in Eq. (27) [51].

$$Path = \{o, L_1(x_1, y_{k1}), L_2(x_2, y_{k2}), \dots, L_{m-1}(x_{m-1}, y_{k(m-1)}), A\}. \quad (27)$$

where $k_i = 1, 2, \dots, 2n + 1$.

The objective function used in single UCAV trajectory planning can thus be defined according to Eq. (28) [51].

$$J = \int_0^{t_f} (\omega_1 \times C^2 + \omega_2 \times h^2 + \omega_3 \times f_{rw}) dt. \quad (28)$$

where C denotes the large cross-track deviations from the line joining the starting and target points, f_{rw} penalizes the penetration trajectories that come dangerously close to the known threat sites, and h minimizes the UCAV's altitude above level h . Since only the horizontal path optimization is taken into account in the study of [51], the objective function is thus simplified to the form shown in Eq. (29) [51].

$$J = L_k + \delta \sum_{i=1}^{m-1} \frac{1}{d_{i\min}}. \quad (29)$$

where L_k is the flight distance, $d_{i\min}$ stands for the distance from the node to the closest threat, δ is an threat avoided coefficient (in general, it is an environmental dependent factor).

The flight distance, from node $a(x_i, y_g)$ in vertical line L_i to node $b(x_{i+1}, y_j)$ in vertical line L_{i+1} can be found in Eq. (30) [51].

$$d_{ab} = \sqrt{\left(\frac{|AB|}{m}\right)^2 + (y_j - y_g)^2}, \quad \text{for } j, g = 1, 2, \dots, 2n+1. \quad (30)$$

Accordingly, the possible UCAV flight distance can be calculated via Eq. (31) [51]:

$$L_k = \sqrt{\left(\frac{|AB|}{m}\right)^2 + (y_{ki} - 0)^2} + \sum_{ki=1}^{m-2} \sqrt{\left(\frac{|AB|}{m}\right)^2 + (y_{k(i+1)} - y_{ki})^2} + \sqrt{\left(\frac{|AB|}{m}\right)^2 + (y_{k(m-1)} - 0)^2}. \quad (31)$$

Suppose that the number of threats is q , and each one of them is described by a circle with the center point denoted by (x_i, y_j) and the radius of r_j , the distance between the $node(x_i, y_{ki})$ and the threat j can be computed through Eq. (32) [51].

$$d = \sqrt{(x_i - x_j)^2 + (y_{ki} - y_j)^2} - r_j. \quad (32)$$

The distance between the $node(x_i, y_{ki})$ and the nearest threat can thus be computed via Eq. (33) [51].

$$d_{imin} = \left[\left\{ \sqrt{(x_i - x_j)^2 + (y_{ki} - y_l)^2} - r_1 \right\}, \dots, \left\{ \sqrt{(x_i - x_q)^2 + (y_{ki} - y_q)^2} - r_q \right\} \right]. \quad (33)$$

Since the generated UCAV optimal trajectory using the proposed IWD algorithm is normally difficult to be implemented in real flying environment due to the potential turning points on the optimized trajectory, the authors of [51] further adopted a class of dynamically feasible trajectory smooth strategy named k-trajectory. Finally, a series of case studies were conducted in their study under complicated combating environments. From the experimental results, it can be observed that the proposed IWD algorithm can find a feasible and optimal trajectory for the single UCAV. Meanwhile, the adopted k-trajectory approach is also every effective in smoothing the UCAV trajectory with a small computational load and real-time simulation possibility.

4.3 Gravitational Search Algorithm (GSA)

Gravitational search algorithm (GSA) was originally proposed by Rashedi, Nezamabadi-pour [52]. In GSA, all the individuals can be mimicked as objects with masses. Based on the Newton's law of universal gravitation, the objects attract each other by the gravity force, and the force makes all of them move towards the ones with heavier masses. In addition, each mass of GSA has four characteristics: position, inertial mass, active gravitational mass, and passive gravitational mass. The first one corresponds to a solution of the problem, while the other three are determined by fitness function. The details of GSA can be summarized as follows.

- First, considering a system with N masses (agents) where the i th mass's position is defined by Eq. (34) [52]:

$$X_i = (x_i^1, \dots, x_i^d, \dots, x_i^n), \quad i = 1, 2, \dots, N. \quad (34)$$

where x_i^d is the position of the i th agent in the d th dimension, and n is the search space's dimension.

- Second, the gravitational force ($F_{ij}^d(t)$) that acting on mass i from mass j at time t can be defined by Eq. (35) [52]:

$$F_{ij}^d(t) = G(t) \frac{M_{pi}(t) \cdot M_{aj}(t)}{R_{ij}(t) + \varepsilon} (x_j^d(t) - x_i^d(t)). \quad (35)$$

where M_{aj} is the active gravitational mass related to agent j , M_{pi} is the passive gravitational mass related to agent i , $G(t)$ is gravitational constant at time t , ε is a

small constant, and $R_{ij}(t)$ is the Euclidian distance between two agents i and j defined by Eq. (36) [52]:

$$R_{ij}(t) = \|X_i(t), X_j(t)\|_2. \quad (36)$$

- Third, for the purpose of computing acceleration of an agent i , total forces (from a group of heavier masses) can be defined by Eq. (37) [52]:

$$F_i^d(t) = \sum_{j=1, j \neq i}^N rand_j F_{ij}^d(t). \quad (37)$$

where $rand_j$ is random number in the interval $[0, 1]$.

- Fourth, based on the total forces, the acceleration of the agent i at time t , and in direction d th, is given by Eq. (38) [52]:

$$a_i^d(t) = \frac{F_i^d(t)}{M_{ii}(t)}. \quad (38)$$

where M_{ii} is the inertial mass of i th agent.

- Fifth, an agent's next velocity can be computed as a fraction of its present velocity added to its acceleration. Both agent i 's new velocity and position are given by Eqs. (39) and (40), respectively [52]:

$$v_i^d(t+1) = rand_i \times v_i^d(t) + a_i^d(t). \quad (39)$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1). \quad (40)$$

where $v_i^d(t)$ and $x_i^d(t)$ are the velocity and the position in d th dimension of agent i at time t , respectively, and $rand_i$ is an uniform random variable in the interval $[0, 1]$ which adds a randomized characteristic to the search.

- Finally, after computing current population's fitness, the gravitational and inertial masses can be updated via Eqs. (41) and (42), respectively [52]:

$$m_i(t) = \frac{fit_i(t) - worst(t)}{best(t) - worst(t)}. \quad (41)$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^N m_j(t)}. \quad (42)$$

where $fit_i(t)$ denotes the fitness value of the agent i at time t , $best(t)$ and $worst(t)$ are defined by Eqs. (43) and (44), respectively [52]:

$$\text{For a minimization problem: } \begin{cases} best(t) = \min_{j \in \{1, \dots, N\}} fit_j(t) \\ worst(t) = \max_{j \in \{1, \dots, N\}} fit_j(t) \end{cases} \quad (43)$$

$$\text{For a maximization problem: } \begin{cases} best(t) = \max_{j \in \{1, \dots, N\}} fit_j(t) \\ worst(t) = \min_{j \in \{1, \dots, N\}} fit_j(t) \end{cases} \quad (44)$$

- Furthermore, to balance the exploration and exploitation of GSA, an agent called k_{best} is employed. It is a function of time which with the initial value k_0 at the beginning and decreasing with time. In GSA, k_0 is normally set to N (i.e., the total number of agents), and k_{best} is decreased linearly. Finally, there will be just one agent applying force to the others. Therefore, the Eq. (37) can be modified as Eq. (45) [52]:

$$F_i^d(t) = \sum_{j \in k_{best}, j \neq i}^N rand_j F_{ij}^d(t). \quad (45)$$

where k_{best} is the set of first k agents with the best fitness value and biggest mass.

- Meanwhile, an initial value of G_0 is always allocated to the gravitational constant, G , which will be reduced with time as defined by Eq. (46) [52]:

$$G(t) = G(G_0, t). \quad (46)$$

The steps of implementing GSA can be summarized as follows [52]:

- Step 1: Determining the system environment.
- Step 2: Randomized initialization.
- Step 3: Fitness evaluation of agents.
- Step 4: Updating the parameters, i.e., $G(t)$, $best(t)$, $worst(t)$, and $M_i(t)$ for $i = 1, 2, \dots, N$.
- Step 5: Calculation of the total force in different directions.
- Step 6: Calculation of acceleration and velocity.
- Step 7: Updating the position of agents.
- Step 8: Repeat Steps 3–7 until the stop criteria is reached. If a specified termination criteria is satisfied stop and return the best solution.

Smart Robot Control using GSA. Although legged robots are slower and more complicated, they have many advantages under certain conditions, such as better adaptability to irregular terrain conditions, and better climbing and obstacle over-crossing capability, which enable them to be more flexible than other types of locomotive mechanisms and can thus be deployed in multitude of dynamically changing situations. While building walking robot, stability is a key factor that needs to be considered since it is fundamental to the overall performance of

terrestrial locomotion. In [53], the author made an attempt to use the characteristics of genetic and GSA to generate gait for a hexapod walking robot. The experimental results demonstrated that, in general, the increase of fitness of transformed gaits can be achieved in comparison with the fitness of the initial gait population. At the end of the study, the author of [53] suggested that supplementary mechanisms can be added for compensating the deviation of the robot path from preset trajectory.

5 Experimental Study

5.1 *Scenario Settings*

Since the main focus of this study falls within AAL robot, we also establish our case study in the context of such theme. It is well known that one of the AAL robots' main applications is in the older person's home, such as healthcare robots [54–58] that are capable of monitoring health (both psychological and medical), detection of falls, medication dispensing, communicating with older people, and assisting physical tasks. Over the decades, the main stream of the existing studies are focused on homogeneous robots (i.e., all individual behaviours are the same), since it becomes easier for one robot to model other robots in the group, which in turn can enhance the whole group's robustness against the failure of individual robot [59]. Nevertheless, there is much to concern the heterogeneous robot groups [60–65] in order to provide more complicated services, such as task assignment, distribution, and rescue. Other examples like robot soccer [66, 67] which involves a variety of agent types, would more easily be solved with a suitable heterogeneous oriented control approach. As a result, this study focuses on a heterogeneous robot group in the AAL environment (i.e., smart home) with each responsible for separate tasks, e.g. cooking, cleaning, maintenance, entertainment, etc. Under this circumstance, when an elderly person suddenly falls down, different functionality equipped robots will form a swarm and try to help by pulling him/her to a comfortable location.

5.2 *Core Challenges Faced by the Experimental Study*

In general, a heterogeneous robot group consists of robots with different designs or functionalities that usually complement each other in order to complete tasks efficiently [68]. Unfortunately, turning such concept (i.e., a heterogeneous swarm of robots) into reality is not very easy. The core challenge for that is how the underlying control schemes will operate, which must deal with the level of behavioural control and the hardware design [68].

Normally, the main inspiration for such swarm robots' behavioural control comes from the observation of social animals (e.g., ants, birds, herd, and fish), since they are capable of self-organizing, decentralization, and emergence of collective behaviour [69]. For example, the behaviour of ants [70] has inspired a number of researchers in collective robotics, such as [71–75]. An interesting initiative is “Swarmanoid” project [68, 76], which provides a modular framework for heterogeneous robot groups' control. Also, a number of researchers (e.g., [65]) have explored heterogeneity at the hardware level (e.g., different sets of sensors or effectors). As such, in this work, we make an attempt to study the effects of a newly developed social animal inspired algorithm (i.e., grey wolf optimizer) in controlling a heterogeneous swarm of AAL robots.

5.3 Employed Novel CI Algorithm

In this case study, we will employ a newly developed algorithm called grey wolf optimizer (GWO) [77] which is inspired by the grey wolves hunting and searching behaviours. In [77], the authors classified the wolves into 4 groups, i.e., alphas, betas, deltas, and omegas. In addition, they assumed that among the groups, alpha (the fittest solution), beta, and delta have better knowledge about the potential location of prey [77]. Overall, the GWO can be seen as a two-stage method, i.e., encircling the prey during the hunting process using hyper-cubes framework and then employing an intensive local search mechanism for optimization. Like many other novel CI algorithms, GWO also includes a balance between exploitation/exploration. This offers the advantage of enhanced search ability while maintaining adequate exploitation capability.

In the following, we describe the steps to be taken for obtaining an efficient implementation of GWO.

- Step 1: Generate the initial the grey wolf population, $X_i (i = 1, 2, \dots, n)$.
- Step 2: Initialize the algorithm parameters (a, A , and C) via Eq. (47) [77]:

$$\begin{aligned}\vec{A} &= 2\vec{a} \cdot \vec{r}_1 - \vec{a} \\ \vec{C} &= 2 \cdot \vec{r}_2.\end{aligned}\tag{47}$$

where $\vec{A}$ and $\vec{C}$ are coefficient vectors, the components of $\vec{a}$ are linearly decreased from 2 to 0 over the course of iterations, and $\vec{r}_1$ and $\vec{r}_2$ are random vectors in $[0, 1]$.

- Step 3: Evaluating the fitness value, i.e., X_α, X_β , and X_δ .
- Step 4: Position correction-cooperation between current search agents by Eqs. 48, and 49, respectively [77]:

$$\begin{cases} \vec{D}_\alpha = |\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}| \\ \vec{D}_\beta = |\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}| \\ \vec{D}_\delta = |\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}| \end{cases} \quad (48)$$

$$\begin{cases} \vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot (\vec{D}_\alpha) \\ \vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot (\vec{D}_\beta) \\ \vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot (\vec{D}_\delta) \end{cases} \quad (49)$$

where $\vec{D}$ is the distance of each candidate solution from the prey, $\vec{X}_\alpha$, $\vec{X}_\beta$, and $\vec{X}_\delta$ are the positions vector of the prey, t represents the current iteration, and $\vec{X}$ indicates the position vector of a grey wolf.

- Step 5: Updating the best location of the hunting wolves through Eq. (50) [77]:

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3}. \quad (50)$$

- Step 6: Evaluating the stopping criteria. If yes, generate output; otherwise, go back to Step 2.

Although the GWO is designed in a very simple manner, i.e., only three main parameters need to be adjusted, each parameter has its own functionalities. For example, the objective for parameters a and A is to find a reasonable balance between a too narrow focus of the search process, which in the worst case may lead to stagnation, and a too weak guidance of the search, which can cause excessive exploration, i.e., when $|A| < 1$, the wolves will attack towards the prey; otherwise, the wolves keep searching for prey. In addition, parameter C has two features: First, it provides random weights for prey in order to stochastically emphasize ($C > 1$) or deemphasize ($C < 1$) the effect of prey in defining the distance; Second, it represents the effect of obstacles to approaching prey in nature [77].

5.4 Testing Results

To solve the focal heterogeneous robot swarm control problem, GWO is utilized to optimize the group performance. We have four kinds of robots, namely, inchworm-type, wheel type, crocodile-type, and caterpillar-type (see Fig. 1 for illustration).

Initially only two types robots were employed and the time of completing the pulling task is recorded. Later on, another type of robot was added and the duration of task finishing is again recorded. Finally, we add one more type of robot into the

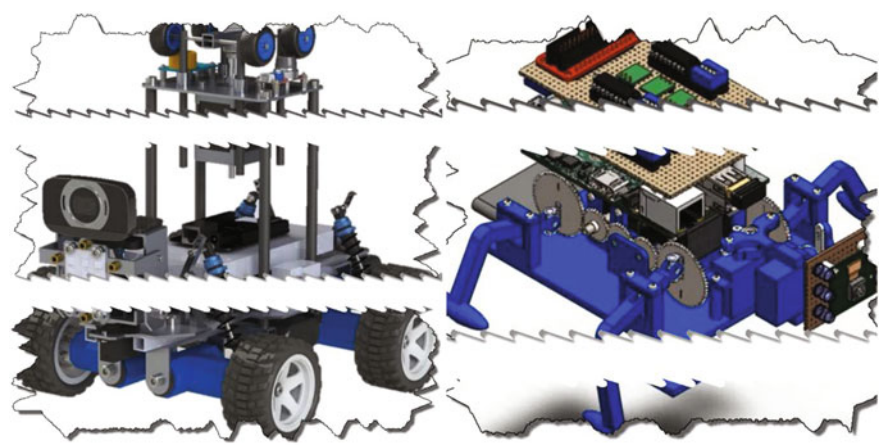


Fig. 1 The schema of wheel-type and crocodile-type robots

No. of Robots	Type of Robots				Time Required (in second)	Distance Traveled (in meter)	Task Accuracy (%)
	Inchworm-type	Wheel-type	Crocodile-type	Caterpillar-type			
2					1200	27	37
2					1600	35	28
2					1100	30	39
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
3					840	45	45
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
4					600	60	75

Legend: denotes the selected robot.

Fig. 2 Preliminary experimental results

recue group and write down the cost of time in this situation. The experimental results are show in Fig. 2. As we can see, different time consumption and task accuracy can be achieved in distinct robot deployment. In general, according to our experiment, the more robots involved, the less time required and the higher task accuracy obtained. Nevertheless, we believe this is scenario dependent and interest readers need to tailor their own arrangement to reach the best overall system performance.

6 Future Work

The limitations of the present work are twofold: On one hand, a widespread survey of the applications of innovative CI presents a difficult task, because of the extensive background knowledge that is required during the process of collection, study, and classification of these publications. Although acknowledging a limited background knowledge, this research makes a brief overview of literature concerned with using biology- and physics-based innovative CI algorithms to deal with robot control. On the other hand, there are also some algorithms that falls under the other categories such as chemistry-, mathematics-based innovative CI. However, the present study does not take them into account. This would be an immediately research direction that need to be considered in future study. Additionally, the presented experimental study is limited to one specific AAL scenario. The future work of this study can be expanded to more general situations with the assistance more comparable novel CI algorithms.

7 Conclusion

Assistive living robot development has for sure not reached its limits and there is still a lot of work need to be done to bridge the gap between academic research and real-world deployment. In this work, we make an attempt to take a quick investigation of several innovative CI techniques and their applications in the area of robot control which is, from the author's perspective, the key to a successfully AAL robot implementation. As the main contribution of this work, we aim at providing a picture about what emerging CI algorithms are being developed and applied to smart robot control area. It is believed that this overview can, through the scattered literature, offer a useful research guide to other scholars who share the similar research interests. The presented demo case also illustrates how these novel CI algorithms can be applied to real world AAL cases.

References

1. Dario, P., et al.: Robotics for medical application. In: *Robotics & Automation Magazine* (1996), pp. 44–56
2. Dario, P., et al.: MOVAID: a personal robot in everyday life of disabled and elderly people. *Technol. Disabil.* **10**, 77–93 (1999)
3. Dario, P., Guglielmelli, E., Laschi, C.: Humanoids and personal robots: design and experiments. *J. Rob. Syst.* **18**(12), 673–690 (2001)
4. JinYoo, S.: Adaptive neural tracking and obstacle avoidance of uncertain mobile robots with unknown skidding and slipping. *Inf. Sci.* **238**, 176–189 (2013)
5. Jaradat, M.A.K., Al-Rousan, M., Quadan, L.: Reinforcement based mobile robot navigation in dynamic environment. *Rob. Comput. Integr. Manuf.* **27**, 135–149 (2011)

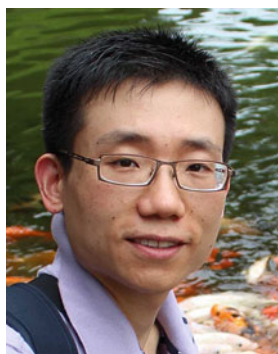
6. Navarro-Guerrero, N., et al.: Real-world reinforcement learning for autonomous humanoid robot docking. *Rob. Auton. Syst.* **60**, 1400–1407 (2012)
7. Maravall, D., Lope, J.D., Domínguez, R.: Coordination of communication in robot teams by reinforcement learning. *Rob. Auton. Syst.* **61**, 661–666 (2013)
8. Herrero-Pérez, D., et al.: Fuzzy uncertainty modeling for grid based localization of mobile robots. *Int. J. Approximate Reasoning* **51**, 912–932 (2010)
9. Miljkovic, Z., et al.: Neural network reinforcement learning for visual control of robot manipulators. *Expert Syst. Appl.* **40**, 1721–1736 (2013)
10. Jolly, K.G., Kumar, R.S., Vijayakumar, R.: Intelligent task planning and action selection of a mobile robot in a multi-agent system through a fuzzy neural network approach. *Eng. Appl. Artif. Intell.* **23**, 923–933 (2010)
11. Motlagh, O., et al.: An expert fuzzy cognitive map for reactive navigation of mobile robots. *Fuzzy Sets Syst.* **201**, 105–121 (2012)
12. Al-Dabbagh, R.D., et al.: System identification and control of robot manipulator based on fuzzy adaptive differential evolution algorithm. *Adv. Eng. Softw.* **78**, 60–66 (2014)
13. Kondo, T., Ito, K.: A reinforcement learning with evolutionary state recruitment strategy for autonomous mobile robots control. *Rob. Auton. Syst.* **46**, 111–124 (2004)
14. Son, J.-H., Choi, Y.-C., Ahn, H.-S.: Bio-insect and artificial robot interaction using cooperative reinforcement learning. *Appl. Soft Comput.* **18**(6), 1127–1141 (2014)
15. Wu, Y.-H., et al.: Designing an assistive robot for older adults: the ROBADMOM project. *IRBM* **34**, 119–123 (2013)
16. Rashidi, P., Mihailidis, A.: A survey on ambient-assisted living tools for older adults. *IEEE J. Biomed. Health Inform.* **17**(3), 579–590 (2013)
17. Feil-Seifer, D., Mataric, M.J.: Defining socially assistive robotics. In: *Proceedings of the 2005 IEEE 9th International Conference on Rehabilitation Robotics*, 28 June–1 July 2005, Chicago, IL, USA
18. Graf, B.: Care-O-bot (2014). <http://www.care-o-bot.de/en/care-o-bot-3.html>. Accessed on 30 July 2015
19. Graf, B., Hans, M., Schraft, R.D.: Care-O-bot II: development of a next generation robotic home assistant. *Auton. Robots* **16**, 193–205 (2004)
20. Graf, B., Parlitz, C., Hägele, M.: Robotic home assistant Care-O-bot 3 product vision and innovation platform. In: Jacko, J.A. (ed.) *Human-Computer Interaction, Part II*, (HCII 2009), LNCS 5611. Springer, Berlin (2009), pp. 312–320
21. RIKEN-TRI Collaboration Center: RIBA (2015) <http://rtc.nagoya.riken.jp/RIBA/index-e.html>. Accessed on 30 July 2015
22. Mukai, T., et al.: Development of a nursing-care assistant robot RIBA that can lift a human in its arms. In: *The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 18–22 Oct 2010, Taipei, Taiwan (2010)
23. Kuindersma, S.R., et al.: Dexterous mobility with the uBot-5 mobile manipulator. In: *International Conference on Advanced Robotics (ICAR)*, June 2009 (2009), pp. 1–7
24. Xu, J., et al.: Enhanced bimanual manipulation assistance with the personal mobility and manipulation appliance (PerMMA). In: *The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 18–22 Oct 2010, Taipei, Taiwan (2010)
25. Wang, H., et al.: The personal mobility and manipulation appliance (PerMMA): a robotic wheelchair with advanced mobility and manipulation. In: *The 34th Annual International Conference of the IEEE EMBS*, San Diego, California USA, 28 Aug–1 Sept 2012 (2012)
26. Cooper, R.A., et al.: Personal mobility and manipulation appliance-design, development, and initial testing. *Proc. IEEE* **100**(8), 2505–2511 (2012)
27. Sato, M., Sugiyama, A., Ohnaka, S.I.: Auditory system in a personal robot, PaPeRo. In: *2006 Digest of technical Papers International Conference on Consumer Electronics (ICCE 06)*, 7–11 Jan 2006 (2006), pp. 19–20
28. Sato, M., et al.: A single-chip speech dialogue module and its evaluation on a personal robot. PaPeRo-mini in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 19–24 April 2009, Taipei, Taiwan (2009), pp. 3697–3700

29. Fujiwara, N., Hagiwara, Y., Choi, Y.: Development of a learning support system with PaPeRo. In: The 12th International Conference on Control, Automation and Systems, 17–21 Oct 2012, Jeju Island, Korea (2012), pp. 1912–1915
30. Hosoda, Y., et al.: Collision-avoidance algorithm for human-symbiotic robot. In: International Conference on Control, Automation and Systems 2010, 27–30 Oct 2010, Gyeonggi-do, Korea (2010), pp. 557–561
31. Hosoda, Y., et al.: Basic design of human-symbiotic robot EMIEW, in Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, 9–15 Oct 2006, Beijing, China (2006), pp. 5079–5084
32. HITACHI. Robotics: EMIEW 2 (2014). http://www.hitachi.com/rd/portal/research/robotics/emiew2_01.html. Accessed on 30 Jul 2015
33. Falconer, J.: HOSPI-R drug delivery robot frees nurses to do more important work (2013). <http://www.gizmag.com/panasonic-hospi-r-delivery-robot/29565/>. Accessed on 30 July 2015
34. Murai, R., et al.: A novel visible light communication system for enhanced control of autonomous delivery robots in a hospital. In: IEEE/SICE International Symposium on System Integration (SII), 16–18 Dec 2012, Kyushu University, Fukuoka, Japan (2012), pp. 510–516
35. Marwala, T., Lagazio, M.: Militarized conflict modeling using computational intelligence. Springer, London, UK (2011). ISBN 978-0-85729-789-1
36. Xing, B., Gao, W.-J.: Innovative computational intelligence: a rough guide to 134 clever algorithms. Springer International Publishing Switzerland, Cham (2014). ISBN 978-3-319-03403-4
37. Passino, K.M.: Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Syst. Manage.* **22**(3), 52–67 (2002)
38. Boussaïd, I., Lepagnot, J., Siarry, P.: A survey on optimization metaheuristics. *Inf. Sci.* **237**, 82–117 (2013)
39. El-Abd, M.: Performance assessment of foraging algorithms vs. evolutionary algorithms. *Inf. Sci.* **182**, 243–263 (2012)
40. Tang, W.J., Wu, Q.H.: Biologically inspired optimization: a review. *Trans. Inst. Meas. Contr.* **31**(6), 495–515 (2009)
41. Hossain, M.A., Ferdous, I.: Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique. *Rob. Auton. Syst.* **64**, 137–141 (2015)
42. Pham, D.T., et al.: The bees algorithm—A novel tool for complex optimisation problems. In: Second International Virtual Conference on Intelligent Production Machines and Systems (IPROMS). Elsevier, Oxford (2006), pp. 454–459
43. Xu, S., et al.: Bio-inspired binary bees algorithm for a two-level distribution optimisation problem. *J. Bionic Eng.* **7**, 161–167 (2010)
44. Krishnanand, K.N., Ghose, D.: Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions. *Swarm Intell.* **3**, 87–124 (2009)
45. Krishnanand, K.N., Ghose, D.: Detection of multiple source locations using a glowworm metaphor with applications to collective robotics. In: IEEE Swarm Intelligence Symposium (SIS). IEEE (2005), pp. 84–91
46. Krishnanand, K.N., et al.: Glowworm-inspired robot swarm for simultaneous taxis towards multiple radiation sources. In: IEEE International Conference on Robotics and Automation (ICRA), May, Orlando, Florida, USA. IEEE (2006), pp. 958–963
47. Birbil, Ş.I., Fang, S.-C.: An electromagnetism-like mechanism for global optimization. *J. Global Optim.* **25**, 263–282 (2003)
48. Wang, Y., et al.: A model predictive control strategy for path-tracking of autonomous mobile robot using electromagnetism-like mechanism. In: International Conference on Electrical and Control Engineering (ICECE). IEEE (2010), pp. 96–100
49. Shah-Hosseini, H.: Problem solving by intelligent water drops. In: IEEE Congress on Evolutionary Computation (CEC), 25–28 Sept 2007. IEEE (2007), pp. 3226–3231
50. Shah-Hosseini, H.: Intelligent water drops algorithm: a new optimization method for solving the multiple knapsack problem. *Int. J. Intell. Comput. Cybern.* **1**(2), 193–212 (2008)

51. Duan, H., Liu, S., Wu, J.: Novel intelligent water drops optimization approach to single UCAV smooth trajectory planning. *Aerosp. Sci. Technol.* **13**, 442–449 (2009)
52. Rashedi, E., Nezamabadi-pour, H., Saryazdi, S.: GSA: a gravitational search algorithm. *Inf. Sci.* **179**, 2232–2248 (2009)
53. Seljanko, F.: Hexapod walking robot gait generation using genetic-gravitational hybrid algorithm. In: 15th International Conference on Advanced Robotics, 20–23 June 2011, Tallinn University of Technology, Tallinn, Estonia. IEEE (2011), pp. 253–258
54. Robinson, H., MacDonald, B., Broadbent, E.: The role of healthcare robots for older people at home: a review. *Int. J. Social Robot.* **6**(4), 575–591 (2014)
55. Broadbent, E., et al.: Attitudes towards health-care robots in a retirement village. *Australas. J. Ageing* **31**(2), 115–120 (2012)
56. Andrade, A.O., et al.: Bridging the gap between robotic technology and health care. *Biomed. Signal Process. Control* **10**, 65–78 (2014)
57. Harrefors, C., Axelsson, K., Sävénstedt, S.: Using assistive technology services at differing levels of care: healthy older couples' perceptions. *J. Adv. Nurs.* **66**(7), 1523–1532 (2010)
58. Broadbent, E., Stafford, R., MacDonald, B.: Acceptance of healthcare robots for the older population: review and future directions. *Int. J. Social Robot.* **1**, 319–330 (2009)
59. Potter, M.A., Meeden, L.A., Schultz, A.C.: Heterogeneity in the coevolved behaviors of mobile robots: the emergence of specialists. In: Proceedings of the 7th International Conference on Artificial Intelligence (2011), pp. 1337–1343
60. Parker, L.E.: ALLIANCE: an architecture for fault tolerant, cooperative control of heterogeneous mobile robots. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (1994), pp. 776–783
61. Haque, M.A.: Biologically inspired heterogeneous multi-agent systems. In: School of Electrical and Computer Engineering, Georgia Institute of Technology (2010)
62. Wurm, K.M., et al.: Coordinating heterogeneous teams of robots using temporal symbolic planning. *Auton. Robots* **34**, 277–294 (2013)
63. Baca, J., Ferre, M., Aracil, R.: A heterogeneous modular robotic design for fast response to a diversity of tasks. *Rob. Auton. Syst.* **60**, 522–531 (2012)
64. Lope, J.D., Maravall, D., Quiñonez, Y.: Response threshold models and stochastic learning automata for self-coordination of heterogeneous multi-task distribution in multi-robot systems. *Robot. Auton. Syst.* **61**, 714–720 (2013)
65. Brunete, A., et al.: A behaviour-based control architecture for heterogeneous modular, multi-configurable, chained micro-robots. *Rob. Auton. Syst.* **60**, 1607–1624 (2012)
66. Stone, P.: Intelligent autonomous robotics: a robot soccer case study. Morgan & Claypool, (2007). www.morganclaypool.com. ISBN 1598291262
67. Lauer, M., et al.: Cognitive concepts in autonomous soccer playing robots. *Cogn. Syst. Res.* **11**, 287–309 (2010)
68. Dorigo, M., et al.: Swarmanoid: a novel concept for the study of heterogeneous robotic swarms. In: IEEE Robotics & Automation (2013), pp. 60–71
69. Dorigo, M., Stützle, T.: Ant Colony Optimization. The MIT Press, Cambridge (2004). ISBN 0-262-04219-3
70. Deneubourg, J.L., et al.: The dynamics of collective sorting robot-like ants and ant-like robots. In: Proceedings of 1st Conference on Simulation of Adaptive Behavior (1991)
71. Kube, C.R., Bonabeau, E.: Cooperative transport by ants and robots. *Rob. Auton. Syst.* **30**, 85–101 (2000)
72. Holland, O., Melhuish, C.: Stigmergy, self-organization, and sorting in collective robotics. *Artif. Life* **5**, 173–202 (1999)
73. Caro, G.D.: A Society of Ant-like Agents for Adaptive Routing in Networks. Universite Libre de Bruxelles, Brussels, Belgium (2002)

74. Dorigo, M.: Swarms of self-assembling robots. In: Weyns, D., Brueckner, S.A., Demazeau, Y. (eds) EEMMAS 2007, LNAI 5049. Springer, Berlin (2008), pp. 1–2
75. Dorigo, M., et al.: SWARM-BOT: design and implementation of colonies of self-assembling robots. In: Yen, G.Y., Fogel, D.B. (eds) Computational Intelligence: Principles and Practice. IEEE Computational Intelligence Society, New York (2006), pp. 103–135
76. Ferrante, E.: A Control Architecture for a Heterogeneous Swarm of Robots: The Design of a Modular Behavior-Based Architecture. Universite Libre de Bruxelles (2009)
77. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. *Adv. Eng. Softw.* **69**, 46–61 (2014)

Author Biography



Bo Xing, DIng, is an Associate Professor at the Department of Computer Science, School of Mathematical and Computer Science, Faculty of Science and Agriculture, University of Limpopo, South Africa. He was a senior lecturer under the division of Center for Asset Integrity Management (C-AIM) at the Department of Mechanical and Aeronautic Engineering, Faculty of Engineering, Built Environment and Information Technology, University of Pretoria, South Africa. Dr. Xing earned his DIng degree (Doctorate in Engineering with a focus on soft computing and remanufacturing) in the early 2013 from the University of Johannesburg, South Africa. He also obtained his B.Sc. and M.Sc. degree both in Mechanical Engineering from the Tianjin University of Science and Technology, P.R. China, and the University of KwaZulu-Natal, South Africa, respectively.

He was a scientific researcher at the Council for Scientific and Industrial Research (CSIR), South Africa. He has published more than 50 research papers in books, international journals, and international conference proceedings. His current research interests lie in applying various nature-inspired computational intelligence methodologies towards big data analysis, miniature robot design and analysis, advanced mechatronics system, and e-maintenance.

Trends in Ambient Intelligent Systems

The Role of Computational Intelligence

Ravulakollu, K.K.; Khan, M.A.; Abraham, A. (Eds.)

2016, X, 234 p. 71 illus., 47 illus. in color., Hardcover

ISBN: 978-3-319-30182-2