The Spread of Innovatory Nature Originated Metaheuristics in Robot Swarm Control for Smart Living Environments

Bo Xing

Abstract The main purpose of introducing ambient assistive living (AAL) robots is to assist the disabled and elderly people at home. In recent years, this field has evolved quickly because of the enormous increase in computing power and availability of the improved variety of sensors and actuators. However, design of AAL robots control system is a huge challenge, which require solving issues related to two classes: design of mechanical structure and development of an efficient control system. In this chapter, we focus on the latter topic, since even relatively low quality hardware can be used for solving sophisticated tasks if the software control it correctly. The chapter starts by giving a vision of what heterogeneous AAL robots is supposed to look like and how a human is to act, navigate and function in it. Particularly, we investigate the effect of artificial neural network (ANN) based control techniques for AAL robots. To enhance the accuracy and convergence rate of ANN, a new method of neural network training is explored, i.e., grey wolf optimization (GWO). Moreover, we provide an overview of applying emerging metaheuristic approaches to various smart robot control scenarios which, from the author's viewpoint, have a great influence on various AAL robot related activities, such as location identification, manipulation, communication, vision, learning, and docking capabilities. The findings of this work can provide a good source for someone who is interested in the research field of AAL robot control. Finally, we concludes with a discussion of some of the challenges that exist in the AAL robot control.

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) · Ambient assisted living (AAL) · Assistive technology services (ATSs) · Nature-inspired computing · Robot control · Metaheuristics

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

H.E. Ponce Espinosa (ed.), *Nature-Inspired Computing for Control Systems*,
Studies in Systems, Decision and Control 40, DOI 10.1007/978-3-319-26230-7_3

1 Introduction

The ability to moving independently is essential for the full development of our lives. Traditionally, for people who have mobility in their upper limbs, the use of mechanical or electric wheelchairs represents a way to regain some of their mobile. However, it is important that the user must acquire knowledge and skills to handle them. In addition, the users have very limited options for take care of themselves. For these reasons, in recent years, there is an increasing interest in the development of assistive technology services (ATSs).

Generally speaking, ATSs aim at applying ambient intelligence technology to support independent living of people with disabilities and special demands [1]. From this perceptive, device-based (e.g., ambient assistive living robots, AAL robots for short) technologies can be seen as important tools that provide high standards to cognitive capabilities, autonomy and movement precision. Applications of AAL robots include independent living [2], surgery [3, 4], rehabilitation [5, 6], and entertainment [7], to name a few. As with any robotics, one of the most challenging problems in AAL robots (in particular with heterogeneous robot members) is the design of an appropriate control system, such as fusion of various sensor information, high accuracy actuation, and reliable software implementation. In this chapter, we intend to utilize artificial neural network (ANN) trained by a novel nature originated metaheuristics for fulfilling such control task.

With this in mind, the remainder of this chapter is organized as follows: First, Sects. 2 and 3 briefly introduce the relevant terms used throughout this study; Second, the focal problem of this chapter is elaborated in Sect. 4 which is followed by a detailed explanation of the employed methodology in Sect. 5; right after this, Sect. 6 describes the experimental environment setup of this chapter; Next the future work provided in Sect. 7 outlines several limitations of the present work; Finally, Sect. 8 draws the conclusion of this study.

2 Background of Ambient Assisted Living (AAL) Robot

According to [8], a robot can be broadly defined as an apparatus that is designed to act as an intelligent linkage between perceiving and acting. Under the umbrella of this definition, mobile robots are mainly referred to the robots the can relocate themselves from one location to another autonomously, i.e., without their surrounding human beings' interference. Unlike their industrial counterparts that can move only along a specific trajectory, mobile robots are capable of moving around freely within a predetermined workspace for the purpose of fulfilling a goal. Following this trend, AAL robot has become an active research in the area of healthcare.

The main purpose of introducing AAL robots is to assist the disabled and elderly people at home [9–11]. Nowadays, there are many types of AAL robots being

developed in research laboratories and by companies all over the world. 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) [10]. For example, "Care-O-bot" [12–14], a robot developed by Fraunhofer IPA, is able to fetch and carry objects, communicate with older people, and supply emergency support; "RIBA" [15, 16], a robot developed by RIKEN-TRI Collaboration Center, can help patient transfer; "uBot5" [17], 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 (i.e., Personal Mobility and Manipulation Appliance)" [18–20], a research outcome from the Carnegie Mellon University and the University of Pittsburgh, can assist persons with severe physical disabilities; "PaPeRo" [21–23], another case developed by NEC, is used to communicate; and "Emiew" [24–26] developed by Hitachi, can interact with human beings; and "Hospi-R" [27, 28], an autonomous delivery robot developed by Matsushita, can even perform complex service tasks. The promising results of the above-mentioned AAL robotic systems indicated that robots could be used as effective ATS tools.

3 Background of Artificial Neural Networks

The original inspiration for ANN is the highly interconnected, massively parallel, distributed, adaptive neuron-and-synapse structure of the brain [29]. In general, the neurons are organized in the form of layers. Each neuron in a layer has weighted connections to neurons in other layers. The main working principle is the weights are adjusted adaptively according to the task under execution in order to improve the overall system performance. According to [30], there are three primary features of ANN, namely utilization of large amounts of sensory information, collective processing capability, and learning and adaptation capability. Broadly, ANN can be classified into two categories: single layer perceptron (SLP) and multi-layer perceptron (MLP).

3.1 Single Layer Perceptron

This network consists of two layers (see Fig.1): the input layer where data or signals are presented to the network, and the output layer which produces the output value of the network to a given input. That means, neurons grouped in layers with only connections between neurons in subsequent layer. Such a network is called a "single-layer", since there is only output layer that performs the computation. It is the basic form of a neural network used for the classification of a special type of

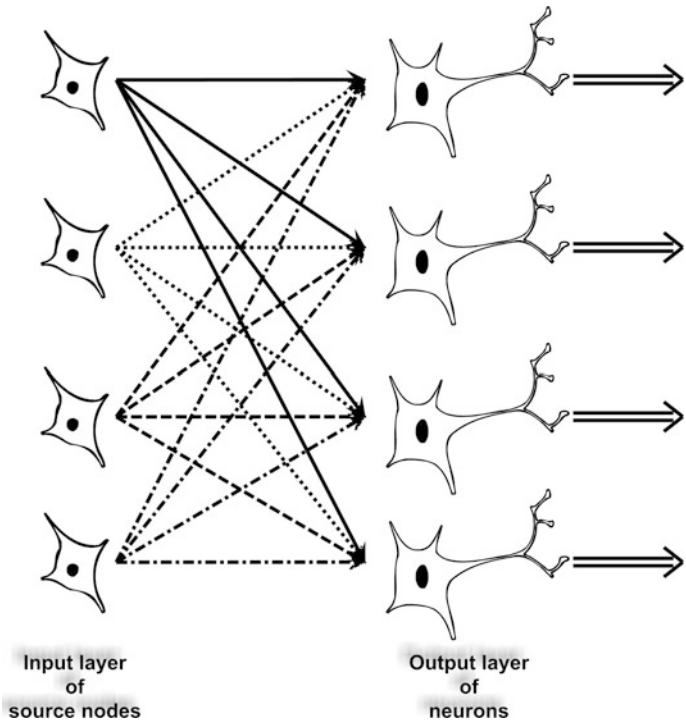


Fig. 1 Feedforward network with a single layer of neurons

patterns, which are linearly separable, i.e., patterns that lie on opposite sides of a hyperplane [31].

In its simplest form, it would consist of set of single inputs to a single neuron, which outputs a single output. A more useful form is illustrated in Fig. 2.

In Fig. 2, a signal x_j at the input of synapse j connected to neuron k is multiplied by the synaptic weight w_{kj}. It also adds the concept of an applied bias, denoted by b_k. It is used to increase or lower the net input of the activation function, depending on whether it is positive or negative, respectively.

In mathematical terms, we may describe the neuron k by writing the pair of equations as shown in Eqs. (1)–(3) [32]:

$$\mu_k = \sum_{j=1}^{m} w_{kj} \cdot x_j \tag{1}$$

$$y_k = \varphi(v_k) \tag{2}$$

$$v_k = \mu_k + b_k \tag{3}$$

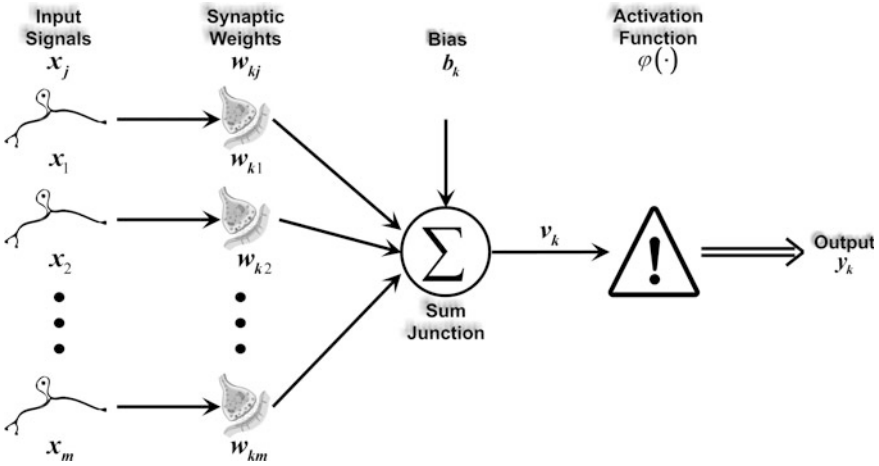


Fig. 2 Nonlinear model of a neuron, labeled k

where $x_1, x_2, \ldots, x_m$ are input signals; $w_{k1}, w_{k2}, \cdots, w_{km}$ are the respective synaptic weights of neuron k; μ_k is the linear combiner output to the input signals; b_k is the bias; $\varphi(\cdot)$ is the activation function which defines the output of a neuron in terms of the induced local field v; v_k is the activation potential or induced local field, and y_k is the output signal of the neuron.

In the simplest case of activation function (i.e., threshold function), the output is computed as Eq. (4) [32]:

$$\varphi(v_k) = \begin{cases} 1 \text{ if } v_k \geq 0 \\ 0 \text{ if } v_k < 0 \end{cases} \tag{4}$$

Correspondingly, the output of neuron k employing such a threshold function is expressed as Eq. (5) [32]:

$$y_k = \begin{cases} 1 \text{ if } v_k \geq 0 \\ 0 \text{ if } v_k < 0 \end{cases} \tag{5}$$

It is important to understand that the form of the activation function, once it is chosen, is will be used for all neurons in the network. In addition, the bias can be seen as an external parameter of neuron k. So we may formulate it as Eq. (6) [32]:

$$v_k = \sum_{j=0}^{m} w_{kj} \cdot x_j \tag{6}$$

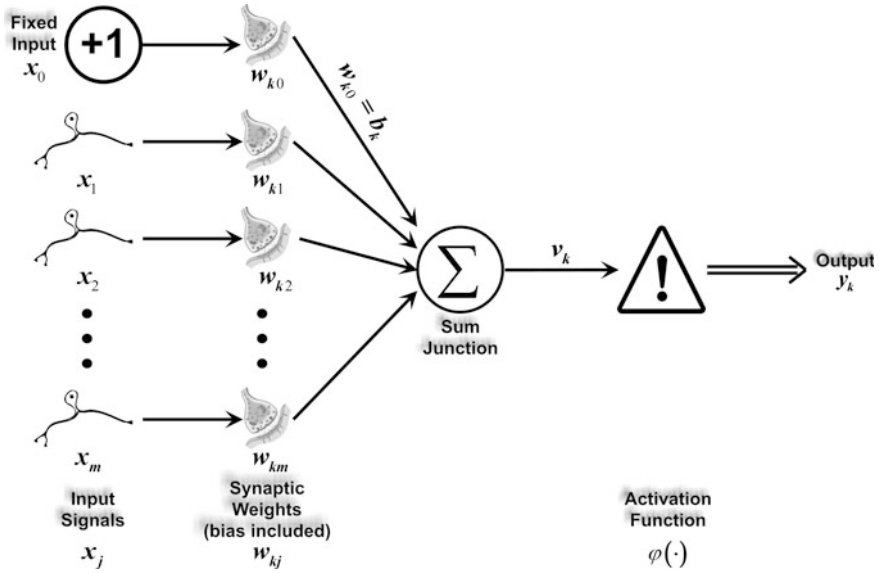


Fig. 3 Another nonlinear model for neuron, labeled k

In Fig. 3 a new synapse is added. Its input is shown in Eq. (7) [32]:

$$x_0 = +1 \tag{7}$$

and its weight is expressed in the form of Eq. (8) [32]:

$$w_{k0} = b_k \tag{8}$$

We may therefore reformulate the model of single neuron k as illustrated in Fig. 3.

3.2 Multi-layer Perceptron

In the literature, a feedforward neural network (FNN) with hidden layers is often referred to as multi-layer perceptron (MLP). The term "hidden" refers to all other layers with no direct connections from or to the outside. The function of hidden neuron is to intervene between the external input and the network output in some useful manner. According to [32], the basic features of MLPs are: (i) there are one or more layers that are hidden from both the input and output nodes; (ii) the network exhibits a high degree of connectivity; and (iii) the model of each neuron in the network has a nonlinear activation function which is differentiable. The performance of the multi-layer neural network is influenced greatly by the number of hidden layers and the number of nodes in a hidden layer. By adding one or more

hidden layers, the network has the capability to extract higher-order statistics from its input. Overall, the output of MLPs are calculated through the following steps [32]:

First, the weighted sums of inputs are computed via Eq. (9) [32]:

$$v_j(n) = \sum_{i=0}^{m} w_{ji}(n) \cdot y_i(n) \tag{9}$$

where $v_j(n)$ is the induced local field produced at the input of the activation function associated with neuron j; m is the total number of inputs (excluding the bias) applied to neuron j; w_{ji} shows the connection weight from the ith node in the input layer to the jth node in the hidden layer; the synaptic weight w_{j0} (corresponding to the fixed input $y_0 = +1$) equals the bias b_j applied to neuron j. Hence, the function signal $y_j(n)$ appearing at the output of neuron j at iteration n is obtained via Eq. (10) [32]:

$$y_j(n) = \varphi_j\big(v_j(n)\big) \tag{10}$$

Then the induced local field for neuron k is expressed in Eq. (11) [32]:

$$v_k(n) = \sum_{i=0}^{m} w_{kj}(n) \cdot y_j(n) \tag{11}$$

where m denotes the total number of inputs (excluding the bias) applied to neuron k; the synaptic weight $w_{k0}(n)$ is equal to the bias $b_k(n)$ applied to neuron k; and the corresponding input is fixed at the value +1.

Next, if neuron j is in the first hidden layer of the network, then $m = m_0$ and the index i refers to the ith input terminal of the network, for which we can write Eq. (12) [32]:

$$y_i(n) = x_i(n) \tag{12}$$

where $x_i(n)$ is the ith element of the input vector (pattern).

On the other hand, if, neuron j is in the output layer of the network, then $m = m_L$ and the index j refers to the jth output terminal of the network, for which Eq. (13) can be obtained [32]:

$$y_j(n) = O_j(n) \tag{13}$$

where $O_j(n)$ is the jth element of the output vector of the MLP. This output is compared with the desired response $d_j(n)$, obtaining the error signal $e_j(n)$ for the jth output neuron.

Finally, the activation function commonly used in MLP is sigmoidal nonlinearity. It is defined via Eq. (14) [32]:

$$\varphi_j\left(v_j(n)\right) = \frac{1}{1+\exp\left(-av_j(n)\right)}, \quad a > 0 \tag{14}$$

where $v_j(n)$ is the induced local field of neuron j, and a is an adjustable positive parameter.

For a neuron j located in the output layer, i.e., $y_j(n) = O_j(n)$, we may express the local gradient for neuron j in the form of Eq. (15) [32]:

$$\begin{aligned}\delta_j(n) &= e_j(n)\cdot\varphi'\left(v_j(n)\right)\\ &= a\cdot\left[d_j(n)-O_j(n)\right]\cdot O_j(n)\cdot\left[1-O_j(n)\right], \quad \text{neuron } j \text{ is an output node}\end{aligned} \tag{15}$$

where $O_j(n)$ is the function signal at the output of neuron j, and $d_j(n)$ is the desired response for it.

On the other hand, for an arbitrary hidden neuron j, we may express the local gradient through Eq. (16) [32]:

$$\begin{aligned}\delta_j(n) &= \varphi'\left(v_j(n)\right)\cdot\sum_k \delta_k(n)\cdot w_{kj}(n)\\ &= a\cdot y_j(\mathrm{n})\cdot\left[1-y_j(n)\right]\cdot\sum_k \delta_k(n)\cdot w_{kj}(n), \quad \text{neuron } j \text{ is hidden}\end{aligned} \tag{16}$$

3.3 *Artificial Neural Network in Robot Control*

Many problems in robotics are unknown, stochastic, and highly non-linear dynamics which offer significant challenges to traditional control methods. To cope with those problems, in recent years, artificial neural network (ANN) based control technique has become more and more popular. A neural network is a massive system of parallel distributed processing elements connected in a graph topology [32]. Usually, the networks are designed to direct a manipulator (e.g., motor) based on sensor data. According to the literature [32–34] the key factors of using ANN control scheme for robotics include: (i) no mathematical process model or rule-based knowledge required; (ii) highly adaptation; (iii) learning capability, such as supervised or unsupervised learning; (iv) fault tolerance; (v) massive parallelism; and (vi) generalization.

Until now, ANN based control has been applied in different robots control areas. For example, the authors of [34] presented a systematic design methodology to a motion adaptive control based on ANN. In [35], a ANN-based controller was

developed for the tracking control of an n rigid-link robot manipulator. In addition, paper [36] employed an observer-based adaptive wavelet neural network tracking control scheme to tackle problems such as system uncertainties, multiple time-delayed state uncertainties, and external disturbances. More recently, a new ANN-based control platform called spiking neural network (SNN) has been used for mobile robot controllers. It used pulse codings to incorporate spatial-temporal information in communication and computation, like real neurons do. Interested readers please refer to [37–39] for more details.

4 Problem Statement

There are many extant versions of ANNs, despite their discrepancies, they all share one common feature that is learning which emphasize an ANN's capability of improving its performance based on the accumulated experience. Briefly, the learning capability of an artificial neuron is achieved by adjusting the weights in accordance to the chosen learning algorithm. Similarly to biological neurons, ANNs have been equipped with mechanisms to adapt themselves to a set of given inputs. Normally, there are two types of learning here: supervised and unsupervised. In the supervised learning, the ANN is provided with feedbacks from an external source (i.e., supervisor). In the latter case, however, an ANN adapts itself to inputs (aka learn) without any extra external feedbacks [40]. In general, the approach that offers learning to an ANN is called trainer. A trainer takes in charge of training NNs to achieve the highest performance for new sets of given inputs. In the supervised learning, a training approach first provides ANNs with a set of data called training data. The trainer then adjusts the structural parameters of the ANN in each training step for the purpose of improving the performance. Once the training phase is accomplished, the trainer is omitted and ANN is ready to use. The trainer if thus often regarded as the most important component of any ANNs.

Typically, there are two types of training approaches in the literature: deterministic versus stochastic. In the first sort, back propagation (BP) and gradient-based methods are often employed. In such techniques, the training phase results in the same performance if the training samples remain consistent. The trainers in this group are mostly mathematical optimization techniques that aim to find the maximum performance (minimum error). In contrast, the trainers in the stochastic camp utilize stochastic optimization techniques to fulfil the goal of maximizing performance of an ANN. The advantages of the deterministic trainers lie in their simplicity and converging speed. Deterministic training algorithm normally starts with a solution and guides it toward an optimum. Though the convergence speed is indeed very fast, quality of the obtained solution highly depends on the initial solution input. Also, there is a high probability of trapping in a local optima which is often referred to the sub-optimal solutions in a search space that might misleading us that the global optimum is obtained. The daunting issue here is

the unknown number of runs that a trainer needs to be restarted with different initial solutions so as to increase the hope of finding global optimum. On the contrary, stochastic trainers start the training process with random solution(s) and evolve it (them). Randomness is the essential component of the stochastic trainers that apply to both initial solutions and method of solution's improvement during the training process. Though they are generally much slower than their deterministic counterparts, the advantage of stochastic methods is the high capability of local optima avoidance. This merit verifies the reasons of why stochastic training methods have been gaining much attention recently [40].

We can roughly divide stochastic training algorithms (in the context of training ANNs) into two groups: single-solution and multi-solution based algorithms. In the first sort, a trainer starts the training with a single randomly constructed ANN and evolves it repeatedly until the stopping criterion are satisfied. Examples of single-solution-based trainers are such as simulated annealing and hill climbing. In the second camp, a multiple-solution-based algorithm initiates the training process by a set of randomly created ANNs and evolves them with respect to each other until the termination conditions are met. The literature indicates that stochastic algorithms with multiple solutions often offer higher ability of escaping from local optima trap. Some of the most popular multi-solution trainers in the literature can be further grouped into two classes: conventional and innovatory metaheuristics [41]. The word heuristic has its origin in the old Greek word *heuriskein*, which means the art of discovering new strategies (rules) to solve problems. The suffix *meta,* also a Greek word, means "upper level methodology." Metaheuristic search approaches can be defined as upper level general methodologies (templates) that can be used as guiding strategies in designing underlying heuristics to solve specific optimization problems [42].

- Conventional metaheuristics: genetic algorithm (GA), particle swarm optimization (PSO), ant colony optimization (ACO), and differential evolution (DE)
- Innovatory metaheuristics: artificial bee colony (ABC), hybrid central force optimization and particle swarm optimization (CFO-PSO), social spider optimization (SSO) algorithm, chemical reaction Optimization (CRO), charged system search (CSS), invasive weed optimization (IWO), and teaching-learning based optimization (TLBO).

The reason as to why these optimization techniques have been employed as training algorithms is their outstanding performance in terms of approximating the global optimum.

To summarize, although many applications of ANNs (e.g., robot control) can be found in the literature, training ANNs is always a challenging task. As we can see from previous discussion, the weights and biases are responsible for defining the final output of MLPs from given inputs. Finding proper values for weights and biases in order to achieve a desirable relation between the inputs and outputs is the core of training MLPs. Bearing this in mind, the focal problem of this chapter is placed in the background of AAL with a focus of training MLPs for the purpose of controlling robot swarm.

5 Employed Methodology

In order to meet the chapter theme, we employ a newly developed algorithm called grey wolf optimizer (GWO) [43] which is inspired by the grey wolves hunting and searching behaviours. In [43], 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 [43]. 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.

5.1 Fundamentals of GWO

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, $i = 1, 2, \ldots, n$.
- Step 2: Initialize the algorithm parameters (a, A, and C) as follows [43]:

$$\begin{aligned} \overrightarrow{A} &= 2\overrightarrow{a} \cdot \overrightarrow{r}_1 - \overrightarrow{a} \\ \overrightarrow{C} &= 2 \cdot \overrightarrow{r}_2 \end{aligned} \tag{17}$$

  where $\overrightarrow{A}$ and $\overrightarrow{C}$ are coefficient vectors, the components of $\overrightarrow{a}$ are linearly decreased from 2 to 0 over the course of iterations, and $\overrightarrow{r}_1$ and $\overrightarrow{r}_2$ 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. (18) and (19), respectively [43]:

$$\begin{cases} \overrightarrow{D}_\alpha &= \left| \overrightarrow{C}_1 \cdot \overrightarrow{X}_\alpha - \overrightarrow{X} \right| \\ \overrightarrow{D}_\beta &= \left| \overrightarrow{C}_2 \cdot \overrightarrow{X}_\beta - \overrightarrow{X} \right| \\ \overrightarrow{D}_\delta &= \left| \overrightarrow{C}_3 \cdot \overrightarrow{X}_\delta - \overrightarrow{X} \right| \end{cases} \tag{18}$$

$$\begin{cases} \overrightarrow{X}_1 = \overrightarrow{X}_\alpha - \overrightarrow{A}_1 \cdot \left(\overrightarrow{D}_\alpha\right) \\ \overrightarrow{X}_2 = \overrightarrow{X}_\beta - \overrightarrow{A}_2 \cdot \left(\overrightarrow{D}_\beta\right) \\ \overrightarrow{X}_3 = \overrightarrow{X}_\delta - \overrightarrow{A}_3 \cdot \left(\overrightarrow{D}_\delta\right) \end{cases} \tag{19}$$

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

- Step 5: Updating the best location of the hunting wolves through Eq. (20) [43]:

$$\overrightarrow{X}(t+1) = \frac{\overrightarrow{X}_1 + \overrightarrow{X}_2 + \overrightarrow{X}_3}{3} \tag{20}$$

- 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 two factors: first, a too narrow focus of the search process, which in the worst case may lead to stagnation; second, 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 [43].

5.2 *Training MLP via GWO*

In general, the following common design questions [42, 44], namely, the representation, the objective, and the evaluation/cost function, are the key to a successful implementation of all metaheuristics. Loosely speaking, the representation refers how a computer keeps the candidate solutions and objects that it handles in the process of searching for new ones; the objective stands for the goal that one is planning to achieve; and the evaluation/cost function denotes a way of verifying the quality of the obtained solution to the problem. In summary, these three pillars form a strategic troika which means one has to give a careful consideration to each element.

The meanings of this statement, in the context of training ANN via metaheuristics, are threefold: first, formulating the problem of training MLP in an acceptable way of utilizing a particular metaheuristic; second, setting up the training purpose, i.e., reducing the difference between the desirable outputs and the obtained outcomes; third, defining a suitable evaluation function that can guide the search of an employed metaheuristic. The following subsections elaborate how this could be done.

5.2.1 Solution Representation

In practice, the alternative candidate solutions are often encoded via representation for manipulating purpose, and thus it is often regarded as one of the fundamental design questions in the development of metaheuristics. For each tackled problem, the search space and its associated size are often determined by how a potential solution is represented and its corresponding interpretations. This statement implies that the efficiency and effectiveness of any implemented metaheuristic are often largely influenced by the chosen representation and the selected way of encoding, rather than by the problem itself. Picking out the right search pace is the first and foremost in implementing a metaheuristic. If an appropriate domain is not targeted at the very beginning of each search, one may fall into two embarrassing situations: either adding a large amount of unfeasible or repeated possible solutions to the shortlist, or excluding the potential right answer(s) from the selected search space.

Although the specific solution representation is often scenario dependent, some general rules can still be concluded as follows [42]:

- Comprehensiveness: A complete set of all possible solutions associated with the tackled problem have to be represented.
- Convexity: There is always a search path exists between any two solutions of the search space.
- Efficacy: The easy-to-manipulate (e.g., time and space complexities) degree of a representation must be high.

In the literature, there are many straightforward encoding methods can be used for dealing with some conventional families of optimization problems (see Fig. 4). In practice, these representation can be combined to form new type of representations for addressing new scenarios.

According to [40], the variables in vector form are suitable for GWO, therefore weights and biases of an MLP are described as Eq. (21) [40]:

$$\mathbf{V} = \{\mathbf{w}, \boldsymbol{\theta}\} = \left\{w_{1,1}, w_{1,2}, \ldots, w_{n,n}, \quad \theta_1, \theta_2, \ldots, \theta_h\right\} \tag{21}$$

where the number of the input nodes is denoted by n, $w_{i,j}$ represents the connection weight from the ith to the jth node, θ_j shows the threshold bias of the jth hidden node.

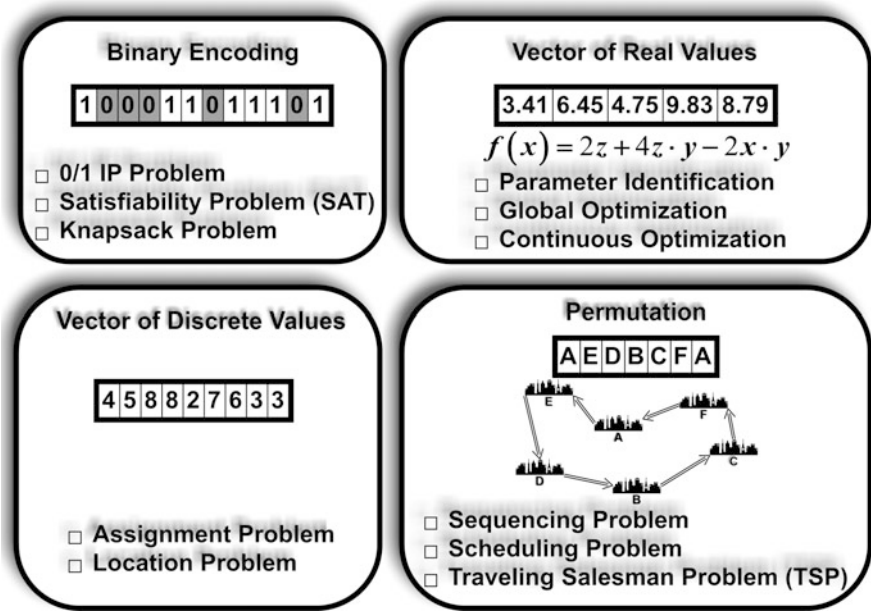


Fig. 4 Typical encoding examples

5.2.2 The Objective

Once the search space has been confirmed after the representation stage, one has to decide what the objective of the targeted problem at hand is. This is more like a task statement (in mathematical form) which concludes what need to be achieved. In the context of MLP training, the main purpose of training an MLP is to achieve the highest accurate degree in terms of classification, approximation, or predication for both training and testing samples. A commonly used measurement for quantifying such achievement is to calculate the mean square error (MSE), which means the difference between the desirable outputs (d_i) and the obtained outcomes (y_i), after a set of training samples being applied to an MLP. The objective of training an MLP can thus be interpreted simply as Eq. (22) [40]:

$$\min \sum (d_i - y_i)^2 \tag{22}$$

Mathematically, Eq. 23 [40] can be employed to calculate such difference.

$$\text{MSE} = \sum_{i=1}^{m} \left(d_i^k - y_i^k\right)^2 \tag{23}$$

where m denotes the number of outputs/outcomes, d_i^k and y_i^k refer to the desired output and actual outcome, respectively, of the ith input node when the kth training sample is applied.

5.2.3 The Evaluation Function

The evaluation function is generally not the same thing as the previously described objective. It is most commonly a mapping from the potential candidate solutions' space (under the selected representation scheme) to a set of numbers (e.g., the reals). In other words, the evaluation function $f(\cdot)$ associates with each candidate solution (belonging to the search space) a numeric value to indicate the quality or the fitness of the solution, e.g., $f : S \rightarrow \mathbb{R}$. The evaluation function gives one an opportunity to compare the usefulness of a present solution with its alternative counterparts. Sometimes, the evaluation function is ordinal since it only offers a ranking of all possible solutions; whereas it may also be numeric by offering not only the order of the solutions but also their corresponding quality degrees [42, 44].

The evaluation function is a crucial element in implementing a metaheuristic since it will guide the search towards "optimal" or "good-enough" solutions within the search space. In a word, regardless of whatever metaheuristic algorithm is employed, no feasible solutions can be obtained if the evaluation function is not properly defined. Nevertheless, in almost all real-world problems, the evaluation function does not come with the problem by itself. How should one go about making such choice? A good rule of thumb is a solution can be regarded as the best evaluation when it meets the objective thoroughly. In other words, it is unacceptable for an evaluation function to conclude that a solution (unable to meet the objective) is better than the other one (successfully meet the objective). But this is too elementary to design a suitable evaluation function.

Often, the objective implies a particular evaluation function. Based on our previous description, an MLP can be regarded as effective only if it is able to adapt itself to the whole training sample set. Accordingly, Eq. (24) below can be used to verify such effectiveness [40]:

$$\overline{\text{MSE}} = \sum_{k=1}^{s} \frac{\sum_{i=1}^{m} \left(d_i^k - y_i^k\right)^2}{s} \tag{24}$$

where s represents the number of training samples, and other parameters follow the same definition described in Eq. (7).

By now, the problem of training an MLP can be summarized and formulated as Eq. (25) below [40]:

$$\text{minimize} : f(\mathbf{V}) = \overline{\text{MSE}} \tag{25}$$

But, what if one cannot always derive valuable evaluation function from the objective? In these cases, one has to be clever enough to resort to some substitute evaluation functions that suit the needs of the problem at hand, the selected representation, and the operators that we implement to go from one solution to the next. The selection of such evaluation function is certainly out of the scope of this chapter, interested readers please refer to [44] for more details.

5.2.4 Training Progress

The generalized process of training an MLP via GWO is depicted in Fig. 5, from which we can see that GWO first takes the $\overline{\text{MSE}}$ from MLP, and then return MLP with the adjusted weights and biases. This process is performed recursively by GWO algorithm where the value of weights and biases are continuously rearranged so that the $\overline{\text{MSE}}$ can be largely minimized for all training data. An expanded version of this training process can be found in [40].

5.3 Concise Summary of Training Performance

As one of the most cited and test datasets, the Iris dataset [45], which comprises 4 attributes, 150 training/test samples, and 3 classes, is also employed to test the performance of training MLP through GWO. Due to the characteristics of Iris dataset, the MLP is constructed as a 4-9-3 structure and the problem thus has 75 variables in total. By testing the GWO-bolstered MLP trainer on the Iris dataset, we can obtain the following experimental results (see Fig. 6 for illustration): $\text{MSE} = 0.229 \pm 0.0032$ and the classification rate equals to 92 %. The results verify the GWO's merits of superior local optimal avoidance capability, and outstanding accuracy achievability. Interested readers please refer to [40] for more evidence of GWO's advanced performance.

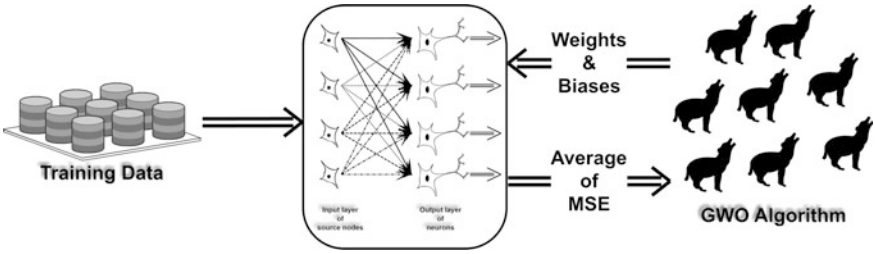


Fig. 5 Flowchart of GWO training MLP [40]

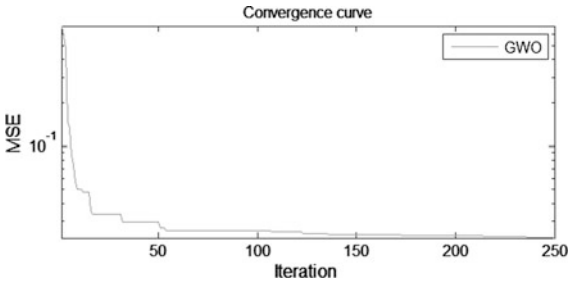


Fig. 6 Converging curve of MLP trained by GWO

6 Experimental Environment Setup

6.1 Focal Scenario

Mobile robots can be roughly classified into the following groups: grounded mobile robots, nautical mobile robots, and aeronautical mobile robots. A complete AAL robot army can certainly involve all these members, say, grounded ones for interaction with elderly people, nautical ones for sewage cleaning, and aeronautical ones for house surroundings surveillance. Nevertheless, this seems too large for our experimental case to handle. So we decide to pay our attention only to the grounded mobile robots with a particular interest in human rescue task. The team members involved in our grouped mobile robots group are legged and wheeled robots. Typically, a heterogeneous robot team comprise distinct robots with different type of designs or functionalities that are most often complementary to each other for the purpose of accomplishing jobs more efficiently [46]. Unfortunately, the realization of a heterogeneous swarm of robots in real-world turns out to be an uphill work. The most daunting issue lies in that how the underlying control scheme could be designed and operated for dealing with the complex behavioural regulation and hardware arrangement [46]. Fortunately, by observing the behaviour of a swarm of social animals (e.g., ants, birds, herd, and fish) such as the features of self-organizing, decentralization, and emergence of collective actions, we can find inspiration of calibrating the desired robot team's control strategy [47]. For instance, the behaviour of ants [48] has inspired a number of researchers in pursuing collective robotics [49–53] design and analysis. An interesting initiative is "Swarmanoid" project [46, 54], which provides a modular framework for heterogeneous robot groups' control. Meanwhile, some researchers (e.g., [55]) also have explored heterogeneity at the hardware level (e.g., different sets of sensors or effectors). Bearing all this ongoing studies in mind, we make an attempt in this chapter to study the effectiveness of employing a novel control scheme, i.e., GWO algorithm trained MLP, in manipulating a heterogeneous swarm of ALL robots.

6.2 Mechatronic Design of Heterogeneous Robot Team

6.2.1 Design Blueprint

An important phase of building any robot swarm is its hardware design which is rather interactive since all components and/or parts will be used to assemble the final prototype. Since most existing work, at the hardware level, have been focusing on exploring the collective behaviour with homogeneous concept. Instead, in this project, we decided to exploit reconfigurability, and modularity using heterogeneous robots with decentralized control algorithms which are influenced by ants, bees colony, and insects behaviours [7]. One of the main characteristics of the designed heterogeneous robot swarm is that team members can perceive their surrounding environmental physical properties via various sensors, and can then executing auto-manipulation and fulfil self-localization through different actuators. The application of the developed robot warm is not only restricted to a research platform but also towards any potential deployment in real-world environments. The modular hardware architecture comprises independent sensory units, actuator modules, and communication kits, which help the swarm system to achieve the scalability and flexibility so that additional sensors and/or actuators could be added without bothering changing the overall architecture.

When designing and implementing the hardware platform for our targeted heterogeneous robot team, there are a lot of issues that need to be taken into account. Some of them are outlined as below:

- For wheel type of robots, a suitable diameter size for the shaft which houses the friction wheels.
- Select right type of wheels for avoiding slipping off while passing through special terrains, e.g., high inclines.
- Due to the rapid development of various techniques such as microcontroller, sensor units, and actuator kits, the platform's upgradability, modifiability, and compatibility need to be carefully considered.
- When integrating modularity and flexibility platform design concepts, one should not sacrifice the feature of user friendly.
- The reconfigurability of the platform should also be enhanced by fully embracing the software and middleware's upgrading.
- A sustainable power consumption plan, say, incorporating a high capacity battery which will lead to less power losses over the circuitry regulation.
- A wireless communication capability is a must for our robot team and thus a low cost but still effective information exchange plan should be investigated for both indoor and outdoor scenarios.
- The factors of size, shape, and weight of the potential platform also require an examination for the sake of robots' movability and manoeuvrability.
- The functionality, coordination, cooperation, and communicability between robots are also worth a scrutiny.

Bearing the abovementioned key points in mind, we set out to construct our heterogeneous robot team. By the time of compiling this chapter, we have preliminary completed two types of living assistant robots (wheel- and crocodile-robot) which are wholly assembled and ready to test for various applications such as simultaneously localization and mapping, avoiding obstacle, performing painting task, and executing rescue duty. The simplicity of the design allows an easy replication of each type of robot which can then form a swarm of robots. By utilizing local information and following some basic rules, these robot agents are able to sense, localize, and actuate for accomplishing complex work. For the rest of this section, the key modules of our robot (including both mechanical and electronic components) are briefed together with their working principles.

6.2.2 Input—Perception

Sensors are vital components of robotics system, since they provide information that allows us to monitor and to control the operation of these systems. Without the availability of sensory information, automated systems cannot operate. In our test environment, the functional movements of a limb involved for achieving a task are generally complex, therefore there is the need of combining different sensor modalities to improve the control process, such as proximity measurement (e.g., ultrasonic and infrared sensors), position measurement (e.g., encoders), temperature measurement (e.g., thermistors), vibration measurement (e.g., accelerometers), chemical measurement (e.g., gas sensors), GPS measurement, and video measurement (e.g., cameras) [56].

- Proximity measurement: Proximity sensors, used to determine the presence of nearby objects, were devised to extend the sensing range beyond that offered by direct-contact tactile sensors [57–59]. In our study, the main purpose of using these types of sensors is to detect obstacle and avoid collision. Based on specific properties used to initiate a switching action, they can be classified into several types: magnetic, inductive, ultrasonic, microwave, optical, and capacitive. Among these, ultrasonic sensors were found to be more accurate and have a much larger detection distance than other types of proximity sensors [59]. In addition, we employ infrared proximity sensors in motion detectors as well.
  - Ultrasonic sensors: Ultrasonic proximity sensors are available with an analog output voltage that is a function of the distance of the object away from the sensor or with two states of digital output that indicate object presence/absence within a defined zone. One feature of ultrasonic sensors is that they are not affected by the colour, transparency, or lighting conditions of the object being detected.
  - Infrared sensors: Infrared sensors consist of two parts: an infrared emitter and receiver. The emitter is actually an LED that emits light that's invisible to the human eye, and the receiver part of the switch collects the IR light that is reflected back. The working principle is that the obstacles cause more light

than usual to be reflected, which tells you that there is something in front of the sensor.

- Position measurement: Position sensors are ones that that provide information about the change in the position of a rigid body [57–59]. These types of sensors can be classified as those that provide analog output (such as potentiometers and resolvers) and those that provide digital output (such as encoders).
  - Digital optical encoders: A digital optical encoder is a non-contact, optical-based device that converts motion into a sequence of digital pulses [60]. They have both linear and rotary configurations. The former are used to measure the linear position and velocity of a translating object, and the latter are used to measure the angular position and velocity for a rotating shaft. According to the literature [61], rotary encoders are the most widely used position sensors in robotic applications, since they provide acceptable resolution with good noise immunity at low cost.
- Temperature measurement: Temperature is a basic quantity in process control systems, and there are several types of sensors available to measure temperature, such as thermistors, thermocouples, resistance temperature detector (RTD), and IC sensors.
- Vibration measurement: Vibratory motion commonly occurs in machinery and flexible structures. Measurement of vibration is important for machine health monitoring of motors, pumps, fans, gearboxes, machine tool spindles, blowers, and chillers. It is usually measured by either accelerometers or vibrometers.
- Chemical measurement: There are many sensors available for detecting chemicals of various kinds, such as smoke, gas, moisture, etc.
  - Gas sensors: Gas sensors contain a small heating element and a catalytic detector to detect concentrations of gases. Using such a device involves supplying a voltage to the heating element and putting the sensors pins in a voltage divider arrangement with a fixed resistor to create a measureable output voltage.
- GPS measurement: GPS relies on a constellation of satellites. Each satellite contains a highly accurate clock that is synchronized with all the other satellites in the constellation. The satellites then broadcast this time signal.
- Cameras: Cameras are considered as complex environmental sensing systems that can be used for capturing and tracking objects locations, recognizing motion patterns, and so on.

6.2.3 Output—Action

In the context of mobile robots, mobility may refer to many actions such as the maximum obstacle size that a mobile robot can get over, the steepest slope that a mobile robot can go up, the largest amount of stairs that a mobile robot can climb,

the deepest swamp that a mobile robot can traverse, the highest distance that a mobile robot can jump, and the widest crevasse that a mobile robot can leap. But all these actions seem to be covered by a more generic concept, that is, mobility system which contains all necessary actuator components for constructing mobile robots. Since wheeled and legged mobile robots are the main focus of this case study, only key mobile system components needed by these robots are elaborated as below.

- Wheel size: The main objective in the control of wheeled robots is to remain balanced and avoid toppling [62]. In general, a bump which the mobile robots climb over, is one-third or less of the diameters of robot's wheels. As a result, to have the enough precession rate for balancing, design parameters of the wheel such as a diameter, a width, a mass, and spinning speed, which determine the size of the wheel that should be selected appropriately.
- Wheel structure: There are many structure alternatives for designing wheeled mobile robots, from more popular type with four-wheels to relatively unusual type with one-wheel. A robot with a single wheel has limited mobility, but enhances its obstacle-crossing ability, smoothness of motion and rolling efficiency [63]. Interested readers please refer to [64] for illustrative examples. For two wheels, there are two obvious layouts, i.e., bicycle-type, and inverted-pendulum type. The former is easier to control at low speeds, but it is not inherently stable; the latter is possible to achieve static stability by accurately placing the center of gravity on the wheel axle, but it always required for dynamic balancing. Three wheels are the minimum required for static stability. They come in many varieties, from very simple two-actuator differential drive, synchronous drive, to complicated holonomic omnidirectional type (e.g., an omnimobile robot with Swedish wheels, active caster wheels, and steerable wheels). Finally, the four wheeled layouts. The well-known one is the car-like structure, i.e., the front two wheels are synchronously steered to keep the same instantaneous center of rotation. A major advantage of this type is that it is stable during high-speed motion. However, it requires a slightly complicated steering mechanism.
- Leg geometry: Like wheeled robots, stability is a major concern in legged robots as well, since they tend to be tall and top heavy. As a result, the types of leg geometries are the key factors for the realization of proper walking capability.
- Leg actuator: One of the greatest problems in the realization of legged robots is the leg actuators which are responsible for the locomotion. In general, there are three major techniques for moving legs on a mobile robot, i.e., linear actuators, direct-drive rotary, and cable driven.

By now, we have our prototype robots (see Fig. 7) designed, built, and ready to be controlled.

6.2.4 Control—Inference

In general, mobile robot design problem can be classified into two directions: design of mechanical structure and development of an efficient control system.

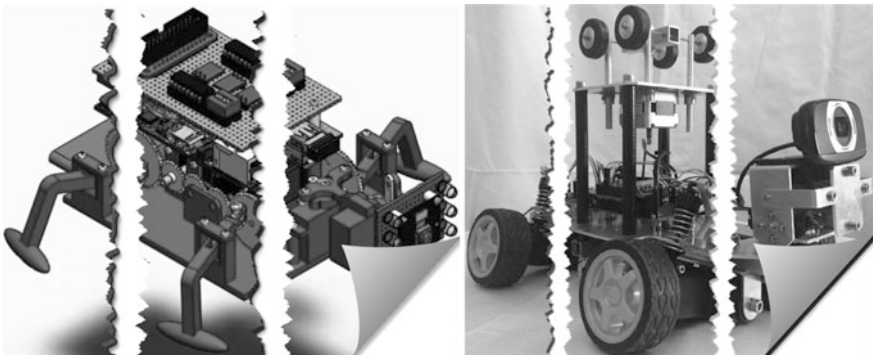
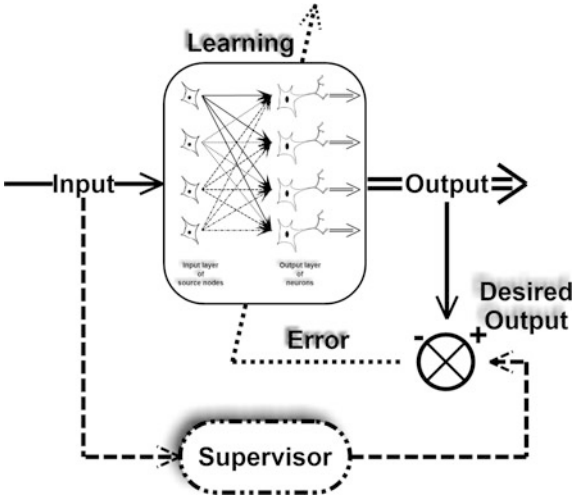


Fig. 7 Prototype robot platform

Typically, robot control systems include open-loop and closed-loop. The architecture of an open-loop system is based on the current state and model of the system, whereas a control system that makes use of feedback is called a closed-loop system. Since the robots usually need to work in an unstructured environment, the closed-loop control system (i.e., with a means of responding to the problems) has been shown more powerful. In particular, if sufficient knowledge can be provided through an autonomous learning process, the control system can result in more autonomous and robust context. Nowadays, computational intelligence algorithms (e.g., artificial neural networks, genetic algorithm, and fuzzy logic) are well known to be computational tools to improve the performance of control techniques. Among others, artificial neural networks (ANNs) worth a mention due to their capability of learning, strong noise toleration, and generalizing. In this chapter, we employ supervised MLP (i.e., multi-layer perceptron) due to its popularity and simplicity (see Fig. 8 for illustration).

Fig. 8 Schematic representation of supervisied MLP



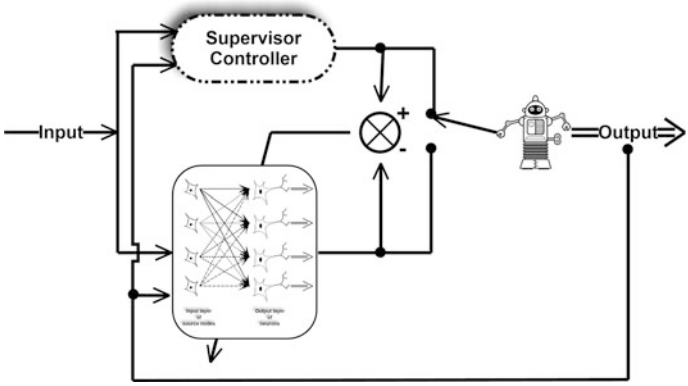


Fig. 9 Structure of neurocontrolled robot with supervised learning

The structure of supervise learning neuron robot control is shown in Fig. 9. As we can see from Fig. 9, the supervisor controller (i.e., GWO in the context of this study) trains the neurocontroller via different training patterns that can control the robot successfully. In detail, during the control period by the supervisor, each neuron is manipulated as a "black box" and all possible combination of neuron input signal and outputs/states of the robotic system are sampled, stored and analysed for the training of the neural network. After the training period, the neurocontroller takes the control actions, and the supervisor is deconnected from the system.

7 Limitations and Future Work

The limitations of the present work are threefold.

7.1 The Implementation of GWO Trained ANN into Heterogeneous Robot Control

In this chapter, we have successfully introduced the GWO trained MLP into our robot swarm control scenario. An immediate future work of the present study is to perform real-world testing on our robot team. The testing is envisioned as below: Initially only MLP controlled robots are employed and the time of completing the pulling task is recorded. Later on, GWO trained MLP control strategy should be introduced and the duration of task finishing is again recorded. For the purpose of evaluation, different time consumption and the achieved task accuracy can be compared between distinct robot deployments. Meanwhile interest readers can tailor their own arrangement to reach the best overall system performance.

7.2 Training ANN via Other Innovatory Nature Originated Algorithms

Recently, there are wide spread of nature originated metaheuristics across the literature. Nevertheless, according to our preliminary overview, the application of this algorithms in training MLP is limited. Apart from several attempts mentioned in the beginning of this chapter, there are still a large amount of approaches left untouched such as bat inspired algorithms, biogeography-based optimization algorithm, cat swarm optimization algorithm, cuckoo inspired algorithms, luminous insect inspired algorithms, fish inspired algorithms, frog inspired algorithms, fruit fly optimization algorithm, group search optimizer algorithm, music inspired algorithms, imperialist competitive algorithm, amoeboid organism algorithm, artificial search swarm algorithm, artificial tribe algorithm, bar systems algorithm, bean optimization algorithm, to name a few. Interested readers please refer to [41] for more details.

7.3 Comparing GWO Trained ANN with Other Versatile Versions of Nature Originated Algorithms in Robot Control

The third limitation of this study is the lack of the comparison between GWO trained ANN and the other nature originated algorithms in terms of robot control. At the end of this study, we briefly overview the corresponding literature with a hope of directing future research in this regard.

7.3.1 Bacteria Foraging Optimization in Robot Control

Bacterial foraging optimization (BFO) algorithm was originally proposed in Passino [65] 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. Based on the basic BFO, an improved version was proposed in [66] 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 [66]. 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.

7.3.2 Bees Algorithm in Robot Control

Inspired by natural foraging behavior of honey bees, the bees algorithm (BA) was proposed by the authors of [67]. 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 [68], 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 [68] to deal with the multi-objective multi-constraint combinatorial optimization task. In BBA, a bee is describe as two binary matrixes **MR** and **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 **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.

7.3.3 Glowworm Swarm Optimization in Robot Control

The glowworm swarm optimization (GSO) algorithm was proposed by the authors of [69]. 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.

In [70], 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 [71], 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 [71] 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.

7.3.4 Electromagnetism-like Mechanism in Robot Control

Similar to that in the elementary electromagnetism, the authors of [72] regarded teach 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.

In order to deal with the path tracking problem, in [73], the authors employed EM to minimize the mobile robot controller's cost function in real time manner. The authors made the comparisons between two algorithm, namely, EM and the 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 [73] 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 [73], the ability to provide an effective and simple predictive control strategy.

7.3.5 Intelligent Water Drops Algorithm in Robot Control

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 [74] invented this algorithm in 2007. 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 [75], the authors made an attempt to solve this imperative task by utilizing IWD algorithm. 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 [75] 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.

7.3.6 Gravitational Search Algorithm in Robot Control

Gravitational search algorithm (GSA) was originally proposed by Rashedi et al. [76]. 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. More detailed discussions regarding GSA can be found in [41, 76].

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 overcrossing 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 [77], 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 [77] suggested that supplementary mechanisms can be added for compensating the deviation of the robot path from pre-set trajectory.

8 Conclusion

The research of AAL robot has for sure not yet reached its frontiers and there is still a lot of work could be done to narrow the gap between academia's know-how and practitioners' requirements. In this work, MLP, an algorithm that is planned to be used for heterogeneous AAL robot team control, was trained via a new proposed GWO algorithm. In order to get desired results, the training problem was first formulated to fit the needs of GWO algorithm. The optimal values for weights and biases of MLP are then determined by GWO. The capability of performing high level of exploration and exploitation proved GWO's usefulness of training MLP. Another contribution made by this study lies in that it also provided a picture about some newly developed nature originated metaheuristic algorithms and how are they being applied to smart robot control area. Though this work is not completed without certain limitations, the study itself is exploratory in nature. It is believed that this chapter can, through the scattered literature, open a new window to other scholars who share the similar research interests.

References

1. Anonymous.: Ambient Assisted Living Roadmap. European Ambient Assisted Living Innovation Alliance. IOS Press, Amsterdam, The Netherland (2010)
2. Borja, R., de la Pinta, J.R., Álvarez, A., Maestre, J.M.: Integration of service robots in the smart home by means of UPnP: a surveillance robot case study. Robot. Auton. Syst. **61**, 153–160 (2013)
3. Schauer, D., Hein, A., Lueth, T.C.: RoboPoint—an autoclavable interactive miniature robot for surgery and interventional radiology. Int. Congr. Ser. **1256**, 555–560 (2003)
4. Pisla, D., Gherman, B., Vaida, C., Suciu, M., Plitea, N.: An active hybrid parallel robot for minimally invasive surgery. Robot. Comput. Integr. Manuf. **29**, 203–221 (2013)
5. Yu, H., Huang, S., Chen, G., Thakor, N.: Control design of a novel compliant actuator for rehabilitation robots. Mechatronics **23**, 1072–1083 (2013)
6. Meng, W., Liu, Q., Zhou, Z., Ai, Q., Sheng, B., Xie, S.S.: Recent development of mechanisms and control strategies for robot-assisted lower limb rehabilitation. Mechatronics (in press)
7. Banks, M.R., Willoughby, L.M., Banks, W.A.: Animal-assisted therapy and loneliness in nursing homes: use of robotic versus living dogs. J. Am. Med. Directors Assoc. **9**, 173–177 (2008)
8. Tzafestas, S.G.: Introduction to Mobile Robot Control. Elsevier Inc., London (2014), ISBN 978-0-12-417049-0
9. Wu, Y.-H., Wrobel, J., Cristancho-Lacroix, V., Kamali, L., Chetouani, M., Duhaut, D., et al.: Designing an assistive robot for older adults: the ROBADOM project. IRBM **34**, 119–123 (2013)

10. Rashidi, P., Mihailidis, A.: A survey on ambient-assisted living tools for older adults. IEEE J. Biomed. Health Inform. **17**, 579–590 (2013)
11. Feil-Seifer, D., Matarić, M.J.: Defining socially assistive robotics. In: Proceedings of the 2005 IEEE 9th International Conference on Rehabilitation Robotics, June 28–July 1, 2005. Chicago, IL, USA (2005)
12. Graf, B.: (2014). Care-O-bot. http://www.care-o-bot.de/en/care-o-bot-3.html. Accessed on 30 July 2015
13. 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)
14. Graf, B., Parlitz, C., Hägele, M.: Robotic home assistant Care-O-bot 3 product vision and innovation platform. In: Jacko JA (ed.) Human-Computer Interaction, Part II, (HCII 2009), LNCS 5611, pp. 312–320. Springer, Berlin (2009)
15. RIKEN-TRI Collaboration Center.: RIBA. http://rtc.nagoya.riken.jp/RIBA/index-e.html. Accessed on 30 July 2015
16. Mukai, T., Hirano, S., Nakashima, H., Kato, Y., Sakaida, Y., Guo, S., et al.: Development of a nursing-care assistant robot RIBA that can lift a human in its arms. In: Presented at the The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 18–22, 2010. Taipei, Taiwan (2010)
17. Kuindersma, S.R., Hannigan, E., Ruiken, D., Grupen, R.A.: Dexterous mobility with the uBot-5 mobile manipulator. In: Presented at the International Conference on Advanced Robotics (ICAR), June 2009, pp. 1–7 (2009)
18. Xu, J., Grindle, G.G., Salatin, B., Vazquez, J.J., Wang, H., Ding, D., et al.: Enhanced bimanual manipulation assistance with the personal mobility and manipulation appliance (PerMMA). In: Presented at the The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 18–22, 2010. Taipei, Taiwan (2010)
19. Wang, H., Grindle, G.G., Candiotti, J., Chung, C., Shino, M., Houston, E., et al.: The personal mobility and manipulation appliance (PerMMA): a robotic wheelchair with advanced mobility and manipulation. In: Presented at the The 34th Annual International Conference of the IEEE EMBS, San Diego, California USA, 28 Aug–1 Sept 2012
20. Cooper, R.A., Grindle, G.G., Vazquez, J.J., Xu, J., Wang, H., Candiotti, J., et al.: Personal mobility and manipulation appliance-design, development, and initial testing. Proceddings IEEE **100**, 2505–2511 (2012)
21. Sato, M., Sugiyama, A., Ohnaka, S.: Auditory system in a personal robot, PaPeRo. In: 2006 Digest of technical Papers International Conference on Consumer Electronics (ICCE 06), pp. 19–20. 7–11 Jan 2006
22. Sato, M., Iwasawa, T., Sugiyama, A., Nishizawa, T., Takano, Y.: A single-chip speech dialogue module and its evaluation on a personal robot, PaPeRo-mini. In: Presented at the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 3697–3700, 19–24 April. Taipei, Taiwan (2009)
23. Fujiwara, N., Hagiwara, Y., Choi, Y.: Development of a learning support system with PaPeRo. In: Presented at the The 12th International Conference on Control, Automation and Systems, pp. 1912–1915, 17–21 October. Jeju Island, Korea (2012)
24. Hosoda, Y., Yamamoto, K., Ichinose, R., Egawa, S., Tamamoto, J.: Collision-avoidance algorithm for human-symbiotic robot. In: Presented at the International Conference on Control, Automation and Systems 2010, pp. 557–561, 27–30 October. Gyeonggi-do, Korea (2010)
25. Hosoda, Y., Egawa, S., Tamamoto, J., Yamamoto, K., Nakamura, R., Togami, M.: Basic design of human-symbiotic robot EMIEW. In: Presented at the Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 5079–5084, 9–15 October. Beijing, China (2006)
26. HITACHI.: Robotics: EMIEW 2. http://www.hitachi.com/rd/portal/research/robotics/emiew2_01.html (2014). Accessed on 30 July 2015
27. Falconer, J.: HOSPI-R drug delivery robot frees nurses to do more important work. http://www.gizmag.com/panasonic-hospi-r-delivery-robot/29565/ (2013). Accessed on 30 July 2015

28. Murai, R., Sakai, T., Kawano, H., Matsukawa, Y.: A novel visible light communication system for enhanced control of autonomous delivery robots in a hospital. In: Presented at the IEEE/SICE International Symposium on System Integration (SII), pp. 510–516, 16–18 December. Kyushu University, Fukuoka, Japan (2012)
29. Russell, S.J., Norvig, P.: Artificial Intelligence: A Modern Approach, 3rd edn. Pearson Education, Inc., Upper Saddle River (2010), ISBN 978-0-13-604259-4
30. Haykin, S.: Neural Networks: A Comprehensive Foundation, 2nd edn. Pearson Education, Inc., Delhi, India, (1999), ISBN 8I-7808-300-0
31. Rosenblatt, F.: Principles of Neurodynamics. Spartan Books, Washington, DC (1962)
32. Haykin, S.: Neural Networks and Learning Machines, 3rd edn. Pearson Education, Inc., Upper Saddle River (2009), ISBN 978-0-13-147139-9
33. Erdem, H.: Application of neuro-fuzzy controller for sumo robot control. Expert Syst. Appl. **38**, 9752–9760 (2011)
34. Patiño, H.D., Carelli, R., Kuchen, B.R.: Neural networks for advanced control of robot manipulators. IEEE Trans. Neural Networks **13**, 343–354 (2002)
35. Wai, R.-J.: Tracking control basedon neural network strategy for robot manipulator. Neurocomputing **51**, 425–445 (2003)
36. Yu, W.-S., Weng, C.-C.: An observer-based adaptive neural network tracking controlof robotic systems. Appl. Soft Comput. **13**, 4645–4658 (2013)
37. Oniz, Y., Kaynak, O.: Control of a direct drive robot using fuzzy spiking neural networks with variable structure systems-based learning algorithm. Neurocomputing **149**, 690–699 (2015)
38. Wang, X., Hou, Z.-G., Zou, A., Tan, M., Cheng, L.: A behavior controller based on spiking neural networks for mobile robots. Neurocomputing. **71**, 655–666 (2008)
39. Wang, X., Hou, Z.-G., Lv, F., Tan, M., Wang, Y.: Mobile robots'modular navigation controller using spiking neural networks. Neurocomputing **134**, 230–238 (2014)
40. Mirjalili, S.: How effective is the grey wolf optimizer in training multi-layer perceptrons. Appl. Intell. **43**, 150–161 (2015)
41. 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
42. Talbi, E.-G.: Metaheuristics: From Design to Implementation. Wiley, Hoboken (2009), ISBN 978-0-470-27858-1
43. Mirjalili, S., Mirjalili, S.M., Lewis, A.: Grey wolf optimizer. Adv. Eng. Softw. **69**, 46–61 (2014)
44. Michalewicz, Z., Fogel, D.B.: How to Solve it: Modern Heuristics, 2nd edn. Springer, Berlin (2004), ISBN 3-540-22494-7
45. Kuncheva, L.: Combining Pattern Classifiers: Methods and Algorithms, 2nd edn. Wiley, Hoboken (2014), ISBN 978-1-118-31523-1
46. Dorigo, M., Floreano, D., Gambardella, L. M.F., Mondada, F., Nolfi, S., Baaboura, T., et al: Swarmanoid: a novel concept for the study of heterogeneous robotic swarms. IEEE Robot. Autom. **20**, 60–71 (2013)
47. Dorigo, M., Stützle, T.: Ant Colony Optimization. The MIT Press, Cambridge (2004), ISBN 0-262-04219-3
48. Deneubourg, J.L., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C., Chretien, L.: The dynamics of collective sorting robot-like ants and ant-like robots. In: Presented at the Proceedings of 1st Conference on Simulation of Adaptive Behavior (1991)
49. Kube, C.R., Bonabeau, E.: Cooperative transport by ants and robots. Robot. Auton. Syst. **30**, 85–101 (2000)
50. Holland, O., Melhuish, C.: Stigmergy, self-organization, and sorting in collective robotics. Artif. Life **5**, 173–202 (1999)
51. Caro, G.D.: A society of ant-like agents for adaptive routing in networks. Unpublished Master Thesis, Universite Libre de Bruxelles, Brussels, Belgium (2002)
52. Dorigo, M.: Swarms of self-assembling robots. In: Weyns D., Brueckner S.A., Demazeau Y. (eds.) EEMMAS 2007, LNAI 5049, pp. 1–2. Springer, Berlin (2008)

53. Dorigo, M., Tuci, E., Trianni, V., Groß, R., Nouyan, S., Ampatzis, C., 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, pp. 103–135. IEEE Computational Intelligence Society, New York (2006)
54. Ferrante, E.: A control architecture for a heterogeneous swarm of robots: the design of a modular behavior-based architecture. Doctor of Philosophy, Universite Libre de Bruxelles (2009)
55. Brunete, A., Hernando, M., Gambao, E., Torres, J.E.: A behaviour-based control architecture for heterogeneous modular, multi-configurable, chained micro-robots. Robot. Auton. Syst. **60**, 1607–1624 (2012)
56. Siegwart, R., Nourbakhsh, I.R.: Introduction to Autonomous Mobile Robots. The MIT Press, Cambridge (2004), ISBN 0-262-19502-X
57. Regtien, P.P.L.: Sensors for Mechatronics. Elsevier Inc., London (2012), ISBN 978-0-12-391497-2
58. Scherz, P., Monk, S.: Practical Electronics for Inventors. McGraw-Hill, New York (2013), ISBN 978-0-07-177134-4
59. Jouaneh, M.: Fundamentals of mechatronics. Cengage Learning, Stamford (2013), ISBN 978-1-111-56901-3
60. Sinclair, I.R., Dunton, J.: Practical Electronics Handbook, 6th edn. Newnes, Elsevier Ltd., Oxford (2007), ISBN 978-0-75-068071-4
61. Kurfess, T.R. (ed.): Robotics and Automation Handbook. CRC Press LLC, Danvers (2005), ISBN 0-8493-1804-1
62. Chan, R.P.M., Stol, K.A., Halkyard, C.R.: Review of modelling and control of two-wheeled robots. Annu. Rev. Control **37**, 89–103 (2013)
63. Xu, Y., Ou, Y.: Control of Single Wheel Robots. Springer, Berlin (2005), ISBN 978-3-540-28184-9
64. Park, J.H., Jung, S.: Development and control of a single-wheel robot: practical mechatronics approach. Mechatronics **23**, 594–606 (2013)
65. Passino, K.M.: Biomimicry of bacterial foraging for distributed optimization and control. IEEE Control Syst. Manage. **22**, 52–67 (2002)
66. Hossain, M.A., Ferdous, I.: Autonomous robot path planning in dynamic environment using a new optimization technique inspired by bacterial foraging technique. Robot. Auton. Syst. **64**, 137–141 (2015) (in press)
67. Pham, D.T., Ghanbarzadeh, A., Koç, E., Otri, S., Rahim, S., Zaidi, M.: The bees algorithm—a novel tool for complex optimisation problems. In: Second International Virtual Conference on Intelligent production machines and systems (IPROMS), pp. 454–459 (2006)
68. Xu, S., Ji, Z., Pham, D.T., Yu, F.: Bio-inspired binary bees algorithm for a two-level distribution optimisation problem. J. Bionic Eng. **7**, 161–167 (2010)
69. Krishnanand, K.N., Ghose, D.: Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions. Swarm Intell. **3**, 87–124 (2009)
70. 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), pp. 84–91 (2005)
71. Krishnan, K.N., Amruth, P., Guruprasad, M.H., Bidargaddi, S.V., Ghose, D.: Glowworm-inspired robot swarm for simultaneous taxis towards multiple radiation sources. In: IEEE International Conference on Robotics and Automation (ICRA), pp. 958–963. May, Orlando, Florida, USA (2006)
72. Birbil, Şİ., Fang, S.-C.: An electromagnetism-like mechanism for global optimization. J. Global Optim. **25**, 263–282 (2003)
73. Wang, Y., Yang, Y., Yuan, X., Yin, F., Wei, S.: 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), pp. 96–100 (2010)
74. Shah-Hosseini, H.: Problem solving by intelligent water drops. In: IEEE Congress on Evolutionary Computation (CEC), pp. 3226–3231, 25–28 September (2007)

75. 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)
76. Rashedi, E., Nezamabadi-pour, H., Saryazdi, S.: GSA: a gravitational search algorithm. Inf. Sci. **179**, 2232–2248 (2009)
77. Seljanko, F.: Hexapod walking robot gait generation using genetic-gravitational hybrid algorithm. In: 15th International Conference on Advanced Robotics, pp. 253–258, 20–23 June. Tallinn University of Technology, Tallinn, Estonia (2011)