SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

February 2022, Vol. 65 122204:1–122204:17
https://doi.org/10.1007/s11432-020-3073-5

# Human guided cooperative robotic agents in smart home using beetle antennae search

Ameer Tamoor KHAN[1], Shuai LI[2*] & Xinwei CAO[3*]

[1]*Department of Computing, The Hong Kong Polytechnic University, Hong Kong* 999077*, China;*
[2]*School of Information Science and Engineering, Lanzhou University, Lanzhou* 730000*, China;*
[3]*School of Management, Shanghai University, Shanghai* 201900*, China*

Received 22 April 2020/Revised 24 July 2020/Accepted 1 October 2020/Published online 20 January 2022

**Abstract** In this paper, we propose a control framework for cooperative robotic agents, which constitutes an essential component in the construction of futuristic smart-homes. Such agents assist humans in efficiently completing household chores. Usability, human friendliness, autonomy, and intelligent decision-making are the top considerations for designing such a system along with reliability, accuracy, and efficiency. Implementing a distributed control algorithm considering these goals is a complicated task since classical control frameworks focus on specialized robots working in an industrial environment and do not capture unique features of the home environment. For example, a household robotic agent needs to perform several general-purpose tasks without assuming that the user has specialized training similar to an industrial operator. Since the challenges and goals in designing a household robotic agent are different, there is a need for a control framework centered around the required goals. The proposed control framework considers the collision problem between several cooperative robotic agents while assisting the human user. We propose an optimization-driven approach to avoid static and dynamic obstacles present in the environment while simultaneously controlling the robots as commanded by the user. We formulate the optimization problem that incorporates the required goals and then use a neural network to solve the optimization problem efficiently. The neural network, beetle antennae search zeroing neural network (BASZNN), is inspired by the natural behavior of beetles. It solves the optimization problem in a gradient-free manner contributing to the computational efficiency of the neural network. Additionally, the distributed-processing capability of the neural networks contributes to computational efficiency and matches the distributed nature of the underlying problem. For testing the performance of BASZNN, we use V-REP and MATLAB to simulate a household environment. Three cooperative agents (KUKA LBR IIWA 7 mounted on P3-DX) assist a person in moving a table around the room. The simulation results show that the BASZNN can accurately and robustly accomplish the required task.

**Keywords** smart-home, assistive agents, metaheuristic optimization, beetle antennae search, zeroing neural network, human guided

**Citation** Khan A T, Li S, Cao X W. Human guided cooperative robotic agents in smart home using beetle antennae search. Sci China Inf Sci, 2022, 65(2): 122204, https://doi.org/10.1007/s11432-020-3073-5

## 1 Introduction

A smart-home refers to an innovative idea where household appliances intelligently interact with each other through the internet. It includes a branch of the omnipresent smart-computing environment, consisting of cooperative robots, to facilitate people within the four walls of their house in a safe, secure, and energy-efficient manner. The connection between the household objects is developed through the vast network of sensors, actuators, and sophisticated computational units embed in those appliances [1]. Smart-homes elevate the standard of life as they allow users to control the appliances at home remotely, and reduce the physical effort. For instance, a person is driving back home from a long hectic day, and in dire need of coffee, he will be able to instruct his coffee maker (at home) from his car to make a coffee for him. Likewise, smart-home monitors and optimizes electric consumption. It also enhances the reliability of the conventional security system with intelligent control.

* Corresponding author (email: Shuaili@ieee.org, xinweicao@shu.edu.cn)

Human assistive agents are a part of smart-homes where they assist users in performing daily chores, e.g., picking and placing items, cleaning, and cooking. However, reliable and robust control of robotic systems are complex tasks, and their inclusion in the human environment requires stringent conditions on the safety and security front. For smart-homes, robots are divided into three categories: social robots, coaching robots, and physical assistive robots [2]. Social robots are exemplified by the Paro robot, which is used for the therapy of older people. It resembles and behaves like a seal to infuse joyful and relaxing emotions in its owner. Likewise, Pepper is another friendly robot that intelligently trains itself according to his owner's emotions and acts accordingly. The second category, the coaching robot. For instance, there are robots for the physical therapy of elderly and partially disabled people; they demonstrate the exercises to them and provide physical support and motivation to engage in it. Likewise, there are robots for the rehabilitation of autism and stroke patients. Similarly, the Pearl robot helps the user to make his schedule and reminds him to perform tasks. The third category is physical assistive robots. Asimo was developed by Honda to assist older people in spoon-feeding, turning off and on the electronic appliances, and picking and placing stuff around the house. Likewise, Robear is another physical assistive robot in the medical field. It has the strength to carry around heavy objects, so it helps the patients move between their beds and the wheelchairs [2].

The control of assistive agents in the smart-home is a complicated task because of its distributed nature. The complexity increases several folds when there is a group of mobile robots, each mounted with a robotic arm. The first objective is the path planning of the assistive mobile agents in a smart-home while avoiding static and dynamic obstacles. The second objective is to use the end-effector of the robotic arm to perform different household chores in the same environment. There are numerous control algorithms proposed for the path planning of the mobile robot in smart-homes. Bevilacqua et al. [3] designed the control problem of the mobile robotic arm, which generates the waypoints from the starting position to the goal destination. Machida et al. [4] used Kinect to gather the 3D position information of humans, and based on that, adjust the mobile robots' velocity and altitude. Chung et al. [5] utilized the single laser range technology to detect and follow the legs of walking humans by mobile robots. Xu et al. [6] proposed a robust omnidirectional path planning of the mobile robot using a neural network (NN). Fierro et al. [7] presented an integrated kinematic controller with a neural network for the path planning of the mobile robot. Likewise, Oubbati et al. [8] used a recurrent neural network (RNN) in two phases for the path planning. In the first phase, it generates the linear and angular velocities for the mobile robot, and in the second phase, it converts those velocities into torque control.

The next challenging task is controlling the redundant robotic arm mounted on an assistive agent. In a redundant robotic arm, the number of joints is more than required for the task, which gives it additional agility and dexterity; however, at the same time, the control gets complex. The literature on the control of redundant robots has discussed several methods to tackle this problem [9–16]. The rudimentary way to address this issue is to linearize and solve the time-varying equations of the redundant robotic arm. The problem with this technique is that it does not generate repeatable results, and it also produces the joint-angle drift. Another well-known conventional method is the use of Jacobian matrix pseudo inverse (JMPI) [17]. The downside of this technique is that it does not work with inequality constraints and can generate undesirable joint-space configurations. Likewise, the calculation of pseudo inverse in the JMPI method is also computationally expensive. Some optimization techniques are also employed to address this issue [18, 19]. Similarly, some advanced techniques, which include NN, fuzzy logic, RNN, and dual neural network (DNN), are also used for redundancy resolution [20–25]. The problems we addressed in this paper are as follows:

- Human guided motion control of mobile agents in a collaborative smart-home environment.
- Avoidance of the static and dynamic obstacles present in a smart-home environment.
- Trajectory control of robotic arms mounted on the mobile platforms to perform the household tasks.

To solve these problems efficiently, we encapsulated them in a single optimization problem, as it gives leverage to achieve any desired goal by properly formulating the optimization problem. The optimization problem includes the motion control of the mobile robot based on its position difference from the target. Furthermore, the optimization problem encapsulates the redundancy resolution of the robotic arm to perform household chores. Lastly, for obstacle avoidance, we incorporated the penalty function to the problem based on maximizing the minimum distance between the obstacles (static or dynamic) and the mobile robot. The penalty function will reward the optimizer on avoiding the obstacle.

In this paper, we use a metaheuristic approach: beetle antennae search zeroing neural network (BASZNN) to solve the optimization problem. A significant benefit of using the proposed metaheuristic

approach is that it allows solving the problem of redundancy resolution on the joint-angle level. Unlike the conventional control algorithms, which address problems at the velocity-level, these methods increase the computational burden caused by pseudo-inversion of the Jacobian matrix in each iteration. Metaheuristic algorithms are known for their efficiency in solving the intricate non-linear and non-convex problems [26]. Our employed metaheuristic approach is a beetle antennae search (BAS) algorithm inspired by the beetle's food searching nature. The successful application of BAS in real-world problems is our motivation to use it to solve the control optimization problem [27–30]. The prediction of confined compressive strength using BAS with neural network [31], BAS for unmanned aerial vehicle (UAV) sensing and obstacle avoidance [32], BAS for the prediction of gases content in transformer oil [33], and economic load distribution using BAS [34] are some of the real-world applications of the BAS algorithm. BAS computes the fitness value of "virtual robots" before computing the objective value of the actual robot. In that case, it gives leverage to feed the applicable states alone to the actual robot. We design the architecture of BAS inspired by zeroing neural network (ZNN) known for efficient zero-searching of non-linear systems. The computational efficiency of ZNN will offer efficient searching of the optimal solution, distributed processing, and accelerated hardware performance, as provided by the intelligent framework implemented in smart-homes. BASZNN has computational, less memory allocation, and time consumption advantages over swarm metaheuristic algorithms, i.e., particle swarm optimization (PSO), ant colony optimization (ACO), firefly algorithm (FA) and genetic algorithms (GA). These algorithms include an army of particles to search the optimal solution with the time and space complexity highly dependent on the number of particles, agents, and the dimension of the agents [35], whereas, BASZNN depends on the dimension of the agents and number of agents, and detailed analysis is provided in Subsection 3.3.2. The highlights of our contribution are as follows.

- We formulated a unified optimization problem to control the cooperative robot in the smart-home environment while considering the problem of obstacle avoidance.
- For analysis of the algorithm, we use V-REP, a robotic simulation tool with MATLAB. We design a smart-home environment with three mobile robots mounted with a redundant robotic arm (KUKA LBR IIWA-14). The task of assisting a person in moving a table while avoiding the static and dynamic obstacles is assigned to the robots to illustrate the proposed algorithm's performance.

The rest of the paper is as follows. In Section 2, we will formulate the optimization problem for assistive robots. In Section 3, we will discuss the framework of BASZNN, its theoretical analysis, and detail time and space complexity. In Section 4, we will demonstrate the smart-home simulation environment in V-REP and the results obtained. In Section 5, we will conclude the paper with final remarks on BASZNN.

## 2 Problem formulation

In this section, we will formulate the problem, the kinematic control of the redundant robotic arm and mobile-base, and the optimization problem for obstacle avoidance. Lastly, we will unify all the sub-optimization problems into one compact and comprehensive problem.

### 2.1 Control of redundant robotic arm in smart-home

Consider a smart-home with robotic agents mounted with a redundant robotic arm on them. To perform an assigned task, the assistive agent will move around in the surrounding, and with the arm, it will reach out for the requested task, for instance, picking a ball. The typical redundant 7-degree of freedom (DOF) robotic agent is shown in Figure 1(a). The kinematic behavior of the robotic arm can be mathematically described using two models: i.e., forward-kinematics (FK) and inverse-kinematics (IK). In FK, angles are input to the joints of the arm, and the output is the coordinates of end-effector in the workspace. The formulation of FK is given as

$$\boldsymbol{X}_{\text{arm}} = f(\boldsymbol{\theta}), \quad \boldsymbol{\theta} = [\theta_1, \theta_2, \theta_3, \ldots, \theta_M]^t, \tag{1}$$

where $\boldsymbol{\theta}$ is a vector containing the angles of all joints. In Figure 1(a), the number of links is seven, so $M = 7$. $\boldsymbol{X}_{\text{arm}} \in \mathbb{R}^N$ represents the coordinates of the end-effector, where in our case the robotic arm is in 3D space, so $N = 3$. As the redundant robotic arm is mounted on a mobile-base, we need to take in account some additional parameters, i.e., base-coordinates and base direction. Mobile-base parameters

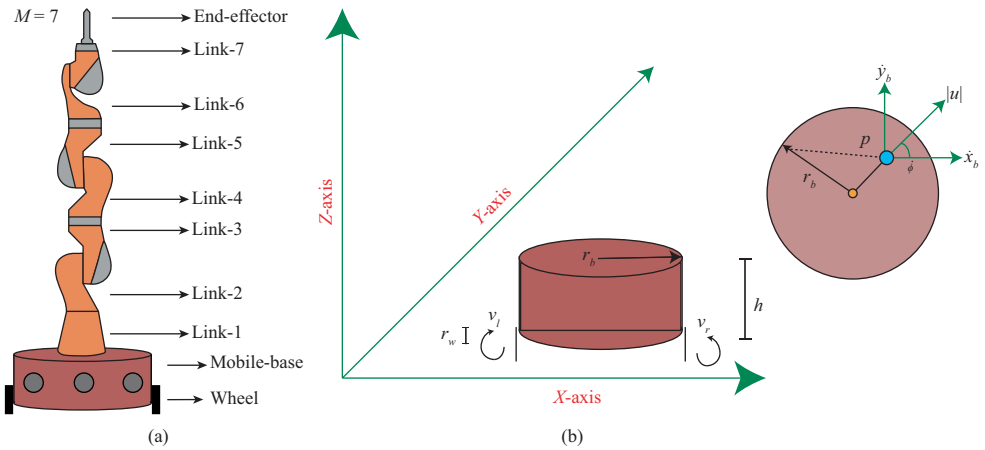



**Figure 1** (Color online) (a) is a well-known 7-DOF redundant robotic arm KUKA LBR IIWA 7 mounted on a Pioneer 3-DX mobile robot. (b) is a schematic of mobile-base with modeling variables ($r_b, r_w$, and $h$), controlling parameters ($\dot{x}_b, \dot{y}_b$, and $\dot{\phi}$), and control signals ($v_l$ and $v_r$).

are given as $\boldsymbol{\Theta} = [x_b, y_b, \phi]$. Now the input to the FK becomes $\boldsymbol{\Phi} = [\boldsymbol{\Theta}, \boldsymbol{\theta}]$. The modified FK of redundant robotic arm is given as

$$\boldsymbol{X}_{\text{arm}} = F(\boldsymbol{\Phi}), \quad \boldsymbol{\Phi} = [x_b, y_b, \phi, \theta_1, \theta_2, \theta_3, \ldots, \theta_M]^t, \tag{2}$$

where $\Phi \in \mathbb{R}^{M+3}$. As the workspace of the end-effector is same, $\boldsymbol{X}_{\text{arm}} \in \mathbb{R}^3$. $F(\cdot)$ is another non-linear transformation which includes the information of the robotic arm with respect to the base. Eq. (2) in terms of its component can be further expressed as follows:

$$\boldsymbol{X}_{\text{arm}} = [x_b, y_b, h] + \Re(\phi) f(\boldsymbol{\theta}), \tag{3}$$

where $x_b, y_b$, and $h$ are the $x$-$y$ coordinates and the height of the mobile-base respectively. $\Re(\phi)$ is the rotation of the base about $z$-axis. However, for real-world applications, FK is not practical; for instance, in a ball picking example, the input should be the coordinates of the ball, and the output should be the angles of the joints required to pick the ball. Here comes the IK, whose formulation is given below:

$$\boldsymbol{\Phi} = F^{-1}(\boldsymbol{X}_{\text{arm}}), \quad \boldsymbol{\Phi} = [x_b, y_b, \phi, \theta_1, \theta_2, \theta_3, \ldots, \theta_M]^t, \tag{4}$$

where $F^{-1}(\cdot)$ is a transformation from workspace to joint-space. As the transformation includes trigonometric function, it is non-linear in nature. Here, we are dealing with redundant manipulators, so it is not only non-linear but highly intricate and complex transformation. Although this is the practical approach, to avoid the computational burden and complexity we will formulate the problem in FK. In the ball picking example, the reference point or the target point is the coordinates of the ball $\boldsymbol{X}_r \in \mathbb{R}^3$. To reach the target point (ball), the robotic arm should follow the generated trajectory such that it ends up at $\boldsymbol{X}_r$. That is how we can avoid the IK computation and devise the trajectory control for the manipulator in FK. The formulation is given below:

$$\boldsymbol{X}_r = F(\boldsymbol{\Phi}_g(t)), \quad \boldsymbol{\Phi}_g = [x_b, y_b, \phi, \theta_1, \theta_2, \theta_3, \ldots, \theta_M]^t, \tag{5}$$

where $\boldsymbol{\Phi}_g(t)$ is the generated trajectory of the robotic arm in joint-space over time $t$. Owing to redundancy and nonlinearity no closed-form solution exists for the redundant robotic arm, so we formulate an optimization problem where the goal is to minimize the error-distance between the coordinates of the end-effector of the robotic arm and the target point. The formulation is given below:

$$G_{\text{arm}}(\Phi_g(t), X_r) = \min_{\Phi_g} ||\boldsymbol{X}_r - F(\boldsymbol{\Phi}_g(t))||_2, \tag{6}$$

where $G_{\text{arm}}(\cdot)$ is the objective function. In Section 4, we will employ our proposed algorithm BASZNN to obtain the optimum solution to this problem.

### 2.2 Control of mobile-base in smart-home

Consider a cylindrical mobile-base with radius $r_b$, height $h$, and the wheel radius $r_w$. These are the modeling parameters of the mobile-base. The mobile-base has two wheels and three controlling parameters: $x_b$, $y_b$, and $\phi$, i.e., coordinates and the direction of mobile-base. They are controlled by two input signals applied to both the wheels (left and right) to control the angular speed, i.e., $v_r$ and $v_l$ of the mobile-base, and it is shown in Figure 1(b). The non-holonomic constraint [36] can be treated as an equality constraint as given below:

$$(\dot{x}_b + p\sin(\phi)\dot{\phi})\sin(\phi) - (\dot{y}_b + p\cos(\phi)\dot{\phi})\cos(\phi) = 0. \tag{7}$$

The addition of equality constraint will reduce one DOF of our system. Let us define the equality constraint in a more compact form as follows:

$$g_{\text{mob}}(\dot{x}_b, \dot{y}_b, \dot{\phi}) = (\dot{x}_b + p\sin(\phi)\dot{\phi})\sin(\phi) - (\dot{y}_b + p\cos(\phi)\dot{\phi})\cos(\phi) = 0. \tag{8}$$

Again, consider the ball picking example. In a home environment, we do not want the robot to stretch its arm before reaching near the ball, as it will be inconvenient for its surroundings. First, the mobile-base will move near the ball, and then its arm will reach out to pick the ball. This leads us towards another reference or target point for the mobile-base $\boldsymbol{X}_r = [x_r, y_r]$. The problem formulation for mobile-base is similar to (6), which is given as

$$G_{\text{mob}}(\boldsymbol{P}_{\text{mob}}, \boldsymbol{X}_r) = \min_{\boldsymbol{P}_{\text{mob}}} \text{dist}(\boldsymbol{X}_r, \boldsymbol{P}_{\text{mob}}), \tag{9}$$

where $\boldsymbol{P}_{\text{mob}} = [x_b, y_b]$ is the current position of mobile-base. The goal is to minimize the distance between the target position and the mobile-base. This concludes the kinematics model for the mobile-base.

### 2.3 Obstacle avoidance in smart-home

The smart-homes are full of household goods, which act as obstacles for the robotic agents. Modern smart-homes are equipped with advanced devices like the camera to observe the surroundings and provide the robotic agents with the positions of obstacles around them [37]. The obstacle can be static (table, chair, and sofa) or dynamic (human and pets). The goal of the robotic agents is to make the 3D map of these obstacles using the sensors and avoid them. Our obstacle avoidance strategy is based on the principle of maximizing the minimum distance between the agent and the obstacle. Here, we will separately formulate the problem of avoiding two components of the robotic system, i.e., for the mobile-base and for the robotic arm. As the obstacles in a home are arbitrarily shaped, the smart-home treats them as a 3D object instead of a point object. The classical techniques to calculate the distance between 3D bodies are computationally expensive and time-consuming. Modern and efficient techniques include a well-known algorithm, Gilbert-Johnson-Keerthi (GJK) [38]. It computes the distance between the vertices of two objects and returns the minimum distance between two vertices. The mobile-base has a uniform cylindrical shape, to avoid extra computation, and we will consider it as a point object. We will calculate the distance from the $z$-axis of the base, where the position of mobile-base is given as $\boldsymbol{P}_{\text{mob}}$. The distance between the obstacle Obs and the mobile-base is given as

$$\text{dist}(\text{Obs}, \boldsymbol{P}_{\text{mob}}) = \text{GJK}(\text{Obs}, \boldsymbol{P}_{\text{mob}}), \tag{10}$$

where $\text{dist}(\cdot)$ is the distance between the obstacle Obs and the mobile-base of our agent. The optimization problem to maximize the minimum distance between obstacle and mobile-base is given below:

$${}^{\text{mob}}G_{\text{obs}} = \frac{1}{\min\limits_{\boldsymbol{P}_{\text{mob}}} \text{dist}(\text{Obs}, \boldsymbol{P}_{\text{mob}})}. \tag{11}$$

The reciprocal of a minimum distance between the two will give the maximum value, i.e., maximize the minimum distance. To avoid overlapping, we need to include a constraint that at least the distance between the two should be greater than the radius of the base $r_b$. Now, the problem becomes

$${}^{\text{mob}}G_{\text{obs}} = \frac{1}{\min\limits_{\boldsymbol{P}_{\text{mob}}} \text{dist}(\text{Obs}, \boldsymbol{P}_{\text{mob}})}$$

$$\text{s.t.} \qquad \text{dist}(\text{Obs}, \boldsymbol{P}_{\text{mob}}) > r_b. \tag{12}$$

Likewise, for redundant robotic arm we need to construct the 3D structure for each link in the robotic arm and calculate its distance from the obstacle. The idea is same to maximize the minimum distance. The GJK distance between the obstacle and links of the robotic arm is given as

$$\text{dist}(\text{Obs}, V_i(\theta_i)) = \text{GJK}(\text{Obs}, V_i(\theta_i)), \tag{13}$$

where $V_i(\cdot)$ represents the vertices of the $i$-th link, i.e., 3D structure. The obstacle avoidance optimization problem for the robotic arm is given below:

$${}^{\text{arm}}G_{\text{obs}} = \frac{1}{\min_{V_i} \text{dist}(\text{Obs}, V_i(\theta_i))}, \quad i \in \{1, 2, \ldots, M\}. \tag{14}$$

To make formulation of obstacle avoidance more clear and compact, we combine (10) and (13) into a single distance formulation which is given as

$$\text{dist}(\text{Obs}, \boldsymbol{P}_{\text{mob}}, V_i(\theta_i)) = \text{GJK}(\text{Obs}, \boldsymbol{P}_{\text{mob}}, V_i(\theta_i)). \tag{15}$$

Now, the modified obstacle avoidance optimization problem for the robotic agent becomes

$$G_{\text{obs}} = \frac{1}{\min_{\text{dist}} \ \text{dist}(\text{Obs}, \boldsymbol{P}_{\text{mob}}, V_i(\theta_i))} \qquad \text{s.t.} \quad \text{dist}(\text{Obs}, \boldsymbol{P}_{\text{mob}}) > r_b. \tag{16}$$

### 2.4 Mechanical constraints of robotic system

The mobile-base has motors installed in the left and right wheels to control the angular speed. Those motors have a mechanical limitation, which does not allow them to go above maximum angular speed. We consider these limitations as an inequality constraint which are given as follows:

$$\dot{v}_r < \dot{v}_{\max}, \quad \dot{v}_l < \dot{v}_{\max}. \tag{17}$$

Likewise, the redundant robotic arm also has a joints-angle limitation for each arm. Any joint can operate between the maximum and minimum allowed angles. Any angle outside the range should be discarded as it can affect the mechanical integrity of the robotic arm. The joints-angle limitation is also treated as an inequality constraint and is given below:

$$\theta_i^+ < \theta_i < \theta_i^-, \quad i \in \{1, 2, \ldots, M\}. \tag{18}$$

### 2.5 Single optimization problem

We have formulated the kinematic model of the agents operating in a smart-home. In our model, we consider the non-holonomic motion planning of mobile robots, obstacle avoidance, and the mechanical limitations of the system. We can combine all the objective functions into one objective function $G(\cdot)$, which is given as

$$G(\cdot) = \min_{\Phi_g} ||\boldsymbol{X}_r - F(\boldsymbol{\Phi}_g(t))||_2 + \min_{\text{dist}} \text{dist}(\boldsymbol{X}_r, \boldsymbol{P}_{\text{mob}}) + \frac{1}{\min_{\text{dist}} \ \text{dist}(\text{Obs}, \boldsymbol{P}_{\text{mob}}, V_i(\theta_i))}. \tag{19}$$

To make the problem more clear and compact we combine together (6), (8), (9), (16)–(18). The unified optimization problem becomes

$$\begin{aligned}
&\gamma_1 \left( \min_{\Phi_g} ||\boldsymbol{X}_r - F(\boldsymbol{\Phi}_g(t))||_2 \right) + \gamma_2 \left( \min_{\boldsymbol{P}_{\text{mob}}} \text{dist}(\boldsymbol{X}_r, \boldsymbol{P}_{\text{mob}}) \right) + \frac{\gamma_3}{\min_{\text{dist}} \ \text{dist}(\text{Obs}, \boldsymbol{P}_{\text{mob}}, V_i(\theta_i))} \qquad (20)\\
&\text{s.t.} \ \ \text{dist}(\text{Obs}, \boldsymbol{P}_{\text{mob}}) > r_b, \ g_{\text{mob}}(\dot{x}_b, \dot{y}_b, \dot{\phi}) = 0,\\
&\dot{v}_r < \dot{v}_{\max}, \ \dot{v}_l < \dot{v}_{\max},\\
&\theta_i^+ < \theta_i < \theta_i^-, \quad i \in \{1, 2, \ldots, M\},\\
&\gamma_1 + \gamma_2 + \gamma_3 = 1, \ 0 \leqslant \gamma_1, \gamma_2, \gamma_3 \leqslant 1,
\end{aligned}$$

where $\gamma_1, \gamma_2$, and $\gamma_3$ are the weights given to three sub-optimization problems. We have included two additional constraints to optimize three weights as well. Now we have three additional variables to optimize. Their optimal value depends on the nature of the task. For example, when the arm reaches out for an object, $\gamma_2$ will have a small value; likewise, when mobile-base is following the reference trajectory, $\gamma_1$ will have a small value. This defines the general framework to operate a single agent in a smart-home environment. In Section 4, we will design a real-world smart-home problem to show these agents working in assistance in smart-home.

## 3 Framework of BASZNN

In this section, we formulate the framework of BASZNN. We then analyze the theoretical aspect to show that it is stable and convergent. Lastly, we discuss in detail the time and space complexity of the algorithm.

### 3.1 BASZNN algorithm

Evolutionary metaheuristic algorithms are nature-inspired techniques to solve optimization problems. BAS is a metaheuristic approach. The inspiration derives from the food searching nature of the beetles. Beetles register the smell of food on its two antennae (left and right) and move in the direction of the intense smell. After repeatedly performing the same task, it ends-up at food. BAS follows the exact logic; at a given position, it moves slightly in the right direction and left direction, and computes the value of the objective function at both directions. The difference of objective function value will decide in which direction the objective function value improves. The BAS moves in that direction and repeats the same procedure in the next iterations until it reaches the optimal solution. The problem with elementary BAS is the intermediary (left and right) evaluation of objective function known as "virtual particles" before actually taking the step. In the case of robotic systems, virtual particles are a mathematical model of the robots. For each configuration of a robot, we need to compute its intermediary states (left and right) to decide which "virtual robot" to follow for the next iteration. It is computationally expensive and time-consuming. To solve this problem, we replaced "virtual particles" with the difference of input and output from their delayed states by the time-factor $\alpha$.

We have implemented BAS in ZNN fashion. ZNN is known for the zero-searching of non-linear systems in a recursive manner using a gradient. ZNN has a deterministic model, which means it is prone to local minima. However, it is computationally more potent than BAS [39]. However, the non-deterministic nature of BAS makes it immune to local-minima. Therefore, we replace the gradient searching of ZNN with BAS and obtain the best of both worlds. The computational power of ZNN will boost the random searching mechanism of the BAS. It will also provide the distributed processing and accelerated performance on hardware, as it is implemented for smart-homes' intelligent framework.

Consider a smart-home environment with assistive agents mounted on with a 7-DOF redundant robotic arm each. Let us say there are $K \in \mathbb{Z}^+$ total agents and at time $t$ the configuration of the $i$-th agent is $\boldsymbol{\Phi}_i(t)$, as shown in (2). The framework to advance the agent in its configuration space $\boldsymbol{\Phi}'_i(t)$ in the next time-step is given below:

$$\boldsymbol{\Phi}'_i(t) = \Lambda(\boldsymbol{\Phi}_i(t) + \lambda_i \boldsymbol{b}_i), \tag{21}$$

where $\boldsymbol{b}_i \in \mathbb{R}^{(M+3)}$, $M$ is a total number of links of the robotic arm which in our case is 7. Here, $b_i(1)$ and $b_i(2)$ correspond to the position of the mobile-base of the $i$-th agent, i.e., $[x_b, y_b]$, $b_i(3)$ is for mobile-base direction $\phi$, and the remaining $\boldsymbol{b}_i$ is for the $M = 7$ links of robotic arm. $\lambda_i$ is a scaling factor, to control the step of the agent in the configuration space $\boldsymbol{\Phi}_i(t)$. Likewise, $\Lambda$ is a projection function that makes sure all the constraints are satisfied. It will discard those inputs that violate the set constraints. As we have several constraints for both mobile-base and redundant arm, we have divided it into three, i.e., $\Lambda_1(\cdot), \Lambda_2(\cdot)$, and $\Lambda_3(\cdot)$. We will see the projection function for all three. $\Lambda_1(\cdot)$ is for the mechanical limitations of the joints-angle constraints (18) and is given as

$$\Lambda_1(\theta_j^i) = \begin{cases} \theta_j^{i-}, & \theta_j^i < \theta_j^{i-}, \\ \theta_j^i, & \theta_j^{i-} < \theta_j^i < \theta_j^{i+}, \\ \theta_j^{i+}, & \theta_j^i > \theta_j^{i+}, \end{cases} \tag{22}$$

where $j$ is the $j$-th link or joint of the $i$-th agent. Likewise, $\Lambda_2(\cdot)$ is for the non-holonomic constraint of the mobile-base (8) and it is given as

$$\Lambda_2(g^i_{\text{mob}}(\dot{x}_b, \dot{y}_b, \dot{\phi})) = 0. \tag{23}$$

Lastly, $\Lambda_3(\cdot)$ is to control the angular velocities of the mobile-base (17) and they are as follows:

$$\Lambda_3(\dot{v}^i_x) = \begin{cases} \dot{v}^i_x, & \text{if } \dot{v}^i_x < \dot{v}^i_{\max}, \\ \dot{v}^i_{\max}, & \text{if } \dot{v}^i_x < \dot{v}^i_{\max}, \end{cases} \tag{24}$$

where $\dot{v}_x$ represents $\dot{v}_r$ and $\dot{v}_l$. All three combine together to form one projection function $\Lambda(\cdot) = \{\Lambda_1(\cdot) \wedge \Lambda_2(\cdot) \wedge \Lambda_3(\cdot)\}$.

Now, we evaluate the objective function value (20), using the new configuration obtained for all $K$ assistive agents in (21). As the purpose is to maintain a cooperative environment, we will add the objective function value (19) of all the agents to evaluate their fitness combine. The formulation is given as

$$\begin{aligned} &H(t) = \sum_{i=1}^{K} G_i(\cdot), \\ &G_i(\cdot) = \gamma_1 \left( \min_{\Phi^i_g} ||\boldsymbol{X}^i_r - F(\boldsymbol{\Phi}^i_g(t))||_2 \right) + \gamma_2 \left( \min_{\boldsymbol{P}_{\text{mob}}} \text{dist}(\boldsymbol{X}^i_r, \boldsymbol{P}^i_{\text{mob}}) \right) + \frac{\gamma_3}{\min_{\boldsymbol{P}_{\text{mob}}} \text{dist}(\text{Obs}, \boldsymbol{P}^i_{\text{mob}}, V^i_j(\theta_j))}, \\ &\quad j \in \{1, 2 \ldots, M\}, \\ &\text{s.t.} \quad \text{dist}(\text{Obs}, \boldsymbol{P}^i_{\text{mob}}) > r_b, \end{aligned} \tag{25}$$

where $G_i(\cdot)$ represents the objective function value of the $i$-th agent, so all the parameters inside the objective function are also according to the $i$-th agent. The summation of all the objective functions $H(t)$ will give the overall cooperative system the fitness value. To overcome the issue of "virtual robots" in traditional BAS, we made two changes. Calculate the difference of objective function $\Delta H$ and the input $\Delta\Phi$ from their respective delayed values by the time-factor $\alpha$. The formulation is given as

$$\Delta H = H(t) - H(t - \alpha), \tag{26}$$

$$\Delta\Phi_i(t) = \boldsymbol{\Phi}'_i(t) - \boldsymbol{\Phi}'_i(t - \alpha). \tag{27}$$

Now to update the configuration state $\boldsymbol{\Phi}^{\text{new}}_i(t)$, it is given as

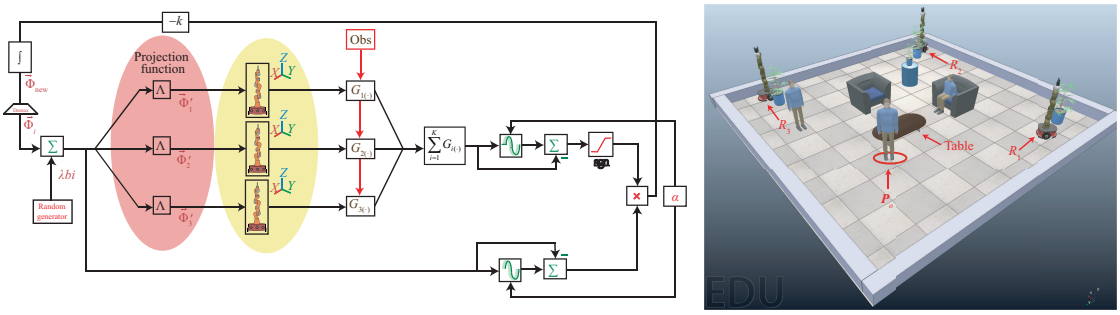
$$\dot{\Phi}^{\text{new}}_i(t) = -k\Delta\Phi_i(t)\text{sgn}(\Delta H), \tag{28}$$

$$\boldsymbol{\Phi}^{\text{new}}_i(t) = -\int_0^t k\Delta\Phi_i(\tau)\text{sgn}(\Delta H)\text{d}\tau, \tag{29}$$

where sgn is a signum function, also known as activation function. It restraints the value of $\Delta H$ within the given limit, i.e., $[-1, 1]$. Now, Eq. (29) runs in closed-loop until $\Delta H = 0$, and the system approaches an optimum solution. The schematic of the proposed BASZNN is shown in Figure 2(a), and the pseudocode is provided in Algorithm 1.

The advantage of BASZNN over its previous variant BAS is the elimination of "virtual particle". As mentioned earlier, BAS computes the objective function value three times on each iteration, making it computationally expensive, slow, and time-consuming for complex systems. From (29) and (26), it can be seen that the update of configuration state $\boldsymbol{\Phi}^{\text{new}}_i(t)$, depends only on the difference of the current and $\alpha$ delayed objective function $\Delta H$. Furthermore, BASZNN has efficient computation and memory utilization than other swarm metaheuristic algorithms, i.e., PSO, ACO, and GA, because of better time and space complexities. These algorithms are composed of an army of particles in search of the optimal solution, whereas, BASZNN is a single particle algorithm backed by the computationally efficient ZNN, which offers robust searching of optimal solution, distributed processing, and accelerated performance of hardware.

Furthermore, our proposed framework works on an optimization driven approach. It does not require any complex mathematical models of the systems included in the smart-home. In this approach, the




(a) (b)

**Figure 2** (Color online) (a) is the BASZNN schematic, and it shows the working framework of the proposed algorithm. It is an integration of beetle antennae search and zeroing neural network, which contains two neural layers. (b) is a smart-home environment we designed in V-REP. It includes three KUKA LBR IIWA 7 (7-DOF robotic arm) mounted on P3-DX (mobile-base) each and person $\boldsymbol{P}_o$ who carries the table with the assistance of agents.

**Algorithm 1** BASZNN pseudocode

**Require:** Kinematic models of the mobile-base and the robotic arm, 3D models of obstacles Obs and the robotic arm's geometry $V_j(\theta_j)$, reference points for robotic agents, i.e., $X_r$, $X_{\text{mob}}$, where $j \in \{1, 2, 3, \ldots, M\}$; same inputs for all agents, i.e., $i \in \{1, 2, 3, \ldots, K\}$;
**Ensure:** Convergence of $H(\cdot) \to 0$, as $t \to t_{\text{end}}$;
**Initialization**
$\alpha \leftarrow 10$; %Delay factor
$t \leftarrow 0$; %Start time
$t_{\text{end}} \leftarrow k$; %End time
$H_{\text{best}} \leftarrow 100$;
**while** $(t < t_{\text{end}})$ **do**
  **for** $(i = 1 : K)$ **do** %$K$ is number of agents
    Generate random direction vector $\boldsymbol{b} \in \mathbb{R}^{M+3}$;
    Compute projection function $\Lambda$ using (22)–(24);
    Compute $\boldsymbol{\Phi}_i'(t)$, using (21);
    Compute $H = H + G_i(\cdot)$, using (25);
    **if** $(H < H_{\text{best}})$ **then**
      $H_{\text{best}} = H$;
      Compute $\Delta\Phi_i(t)$, using (27);
      **if** $(i == K)$ **then**
        Compute $\Delta H$, using (26);
        **for** $(i = 1 : K)$ **do**
          Compute $\boldsymbol{\Phi}_i^{\text{new}}(t)$, using (29);
        **end for**
      **end if**
      **if** $(\text{mod}(t, (t - t_{\text{end}})) == \alpha)$ **then**
        $H(t - \alpha) = H$;
        $\boldsymbol{\Phi}_i'(t - \alpha) = \boldsymbol{\Phi}_i'(t)$;
      **end if**
    **end if**
  **end for**
**end while**

robots have no active communication among themselves. Instead, the algorithm works as a central-processing-unit, which receives data from all the robots, computes them, and sends the data back to them such that they work cooperatively, i.e., under given constraints.

### 3.2 Theoretical analysis of BASZNN

Here, we will analyze BASZNN theoretically, and discuss its stability and convergence.

**Lemma 1.** BASZNN is stable as the objective function value $G(\cdot)$ decreases monotonically with time $t$. The stability of BASZNN can be described as follows:

$$G_{t_1}(\cdot) > G_{t_2}(\cdot), \quad t_1 < t_2. \tag{30}$$

*Proof.* Zhang et al. [40] provided a detail proof of Lemma 1.

**Lemma 2.** BASZNN is convergent as when time $t \to \infty$, the assistive agents converge to their reference or target point. The convergence of BASZNN can be described as follows:

$$G_{\text{arm}}(t) \to X_r, \qquad t \to \infty, \tag{31}$$

$$G_{\text{mob}}(t) \to X_{\text{mob}}, \quad t \to \infty. \tag{32}$$

As the mobile-base and redundant robotic arm eventually converge to their respective reference point, we can infer from (35) and (36) that,

$$G(t) \to 0, \quad t \to \infty. \tag{33}$$

*Proof.* Zhang et al. [40] provided a detail proof of Lemma 2.

### 3.3 Time and space complexity

Here we discuss the time and space complexity of BASZNN. Algorithm 1 shows the framework of BASZNN algorithm. From there, we can evaluate the time and space complexity of the algorithm.

#### 3.3.1 *Time complexity*

As the algorithm runs for $t_{\text{end}}$ times, let us say, $t_{\text{end}} = N \in \mathbb{Z}$, which means it has a time complexity of $N$. Next, we have a "for" loop that runs $K$ times (total number of agents), so it has the time complexity of $K$. Within "for" loop, we have a random direction vector $\boldsymbol{b}$ with time complexity of $M+2$. Next is the $\Lambda$ function. For $\Lambda_1$ the complexity is $M$, and for $\Lambda_2$ and $\Lambda_3$ the complexity is 1. The total time complexity of $\Lambda$ is $M+2$. Then the time complexity of $G_i(\cdot)$ is $2M+2$, as from (20) we can see it involves two objective functions for $M$-DOF redundant robotic arms, and two for the mobile-base. Next, the computation of $\boldsymbol{\Phi}_i(t)$ also has a complexity of $M+3$. Likewise, $\Delta H$ has a complexity of 1. Then, BASZNN has another "for" loop with the complexity of $K$. Within the "for" loop, there is $\boldsymbol{\Phi}_i(t)$, which again has the complexity of $M+3$. Finally, $H(t-\alpha)$ and $\boldsymbol{\Phi}_i'(t)$ have the complexity of 2. If we combine the total complexities, it becomes $N[K[(M+2)+(M+2)+(2M+2)+(M+3)+1+K(M+3)+2]] = N[K[(K+6)M+12]]$. Further simplification shows, $N[K^2M+6KM+12K] = NK^2M+6NKM+12NK$, as the time $N$ is linear so we can discard it and the total time complexity becomes $\mathcal{O}(MK^2)$, where $K$ is the total number of agents in a smart-home environment, and $M$ is the dimension.

#### 3.3.2 *Space complexity*

For the space complexity, we alone consider those space allocations, which constitute the most of the memory. We can see that there are two "for" loops, and within the second loop the memory consumes by $\boldsymbol{\Phi}_i(t)$ is $M+3$, so the space complexity for this alone becomes $\mathcal{O}(MK^2+3K)$, which is of order $\mathcal{O}(MK^2)$. The rest of the space allocation is of degree one, which is less than $K^2$.

## 4 Simulation environment and results

In this section, we use BASZNN to control the assistive agents in a smart-home environment. Then we analyze the results in detail.

### 4.1 Smart-home environment

Smart-homes are fully equipped with sensors and devices to control and monitor all household equipment and feed them with necessary instructions and user commands to perform certain tasks. The controlling and monitoring system works as the central-brain of the smart-homes. In our case, an application of assistive agents is implemented to test the working of the BASZNN algorithm. The smart-home environment we used in our case is shown in Figure 2(b). It includes household commodities like sofas, tables, and plant pots. It also includes three assistive agents, named as $R_1, R_2$, and $R_3$. The environment also includes humans walking freely around the living room. The scenario is as follows: a person wants to lift the table and move it to another part of the room. The table is heavy, so he requires assistance. The person will request for assistance to the central controlling unit of the smart-home. The central system will direct the agents toward the task and provide them with the necessary information, including all the goal positions of robotic agents $\boldsymbol{X}_r^i$, $i \in \{1, 2, \ldots, K\}$.

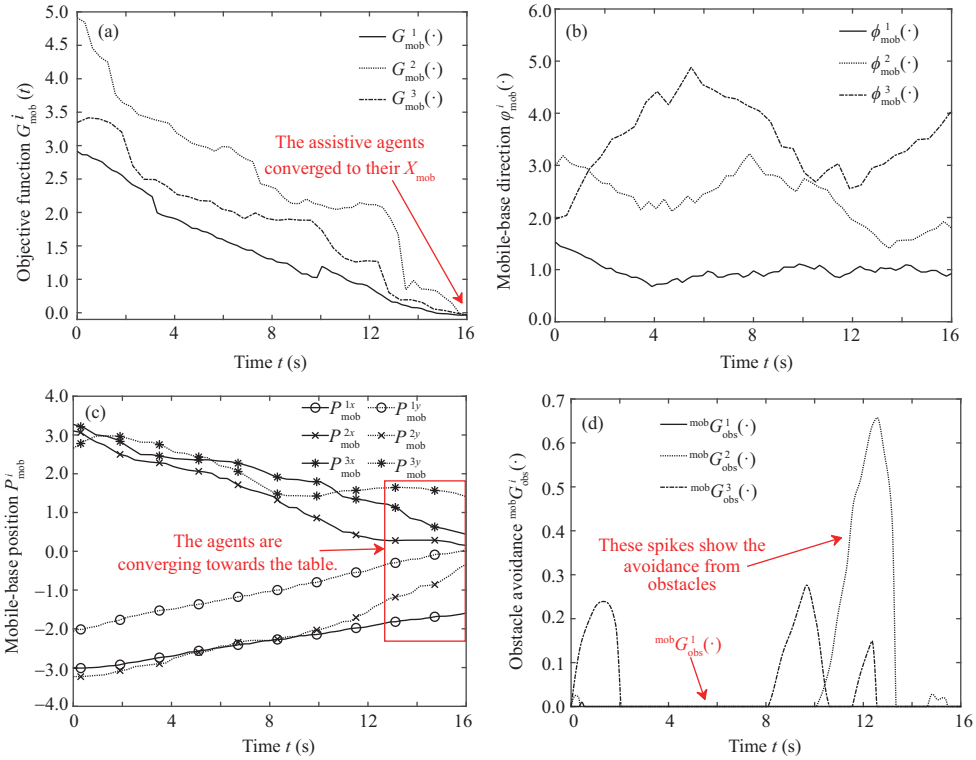



**Figure 3** (Color online) The "approaching" phase of the simulation. (a) shows the convergence of the objective function of all three agents. (b) shows the direction of agents while moving from their home position towards the table; likewise, (c) shows their positions. Lastly, (d) shows the objective function values for the avoidance of the obstacles (static and dynamics).

### 4.2 Constraints of the task

As earlier, we have discussed the constraints for the control of robotic agents in detail, but this specific task includes some additional constraints. For instance, when robotic agents pick the table, their grip on the table should remain intact. Otherwise, they can cause scratches on the table, or the table may fall off owing to the loose grip. To accommodate these limitations, we included some additional constraints. In our case we used three assistive robots; let us say, the gripping positions on the table of all three agents are $P_i \in \mathbb{R}^3$, where $i = [1, 2, 3]$ and the gripping position of the person is $P_o \in \mathbb{R}^3$. There is a total of two additional conditions that we need to consider, which are given below.

- The table should remain stable while moving from one place of the room to the other.
- The table should neither stretched nor pressed, i.e., a constant distance between all grips.

First, the stability of the table is possible if the height of all the robotic arms while holding and carrying the table remains constant. The formulation of the first constraint is given as

$$Z(P_k) = c, \qquad k \in \{o, 1, 2, \ldots, K\}, \tag{34}$$

where $Z(\cdot)$ is a height function; it computes height at point $P_i$. $K$ here shows the number of agents involved to perform the task, and $o$ is for the grip of the person. The height $Z(\cdot)$ of all the gripping points should remain constant $c$. Secondly, to avoid the stretching and pressing of the table, we divided the problem in two parts, i.e., the constant distance between $P_i$ and $P_o$, and the constant distance between all $P_i$. The first subdivided problem is given as

$$||P_i - P_o|| = d_{io}, \qquad i \in \{1, 2, \ldots, K\}, \tag{35}$$

where $d_{io}$ is the distance between the agent $P_i$ and the person $P_o$. The formulation of the second subdivided problem is

$$||P_1 - P_2|| = d_{12}, \quad ||P_1 - P_3|| = d_{13}, \quad ||P_2 - P_3|| = d_{23}. \tag{36}$$

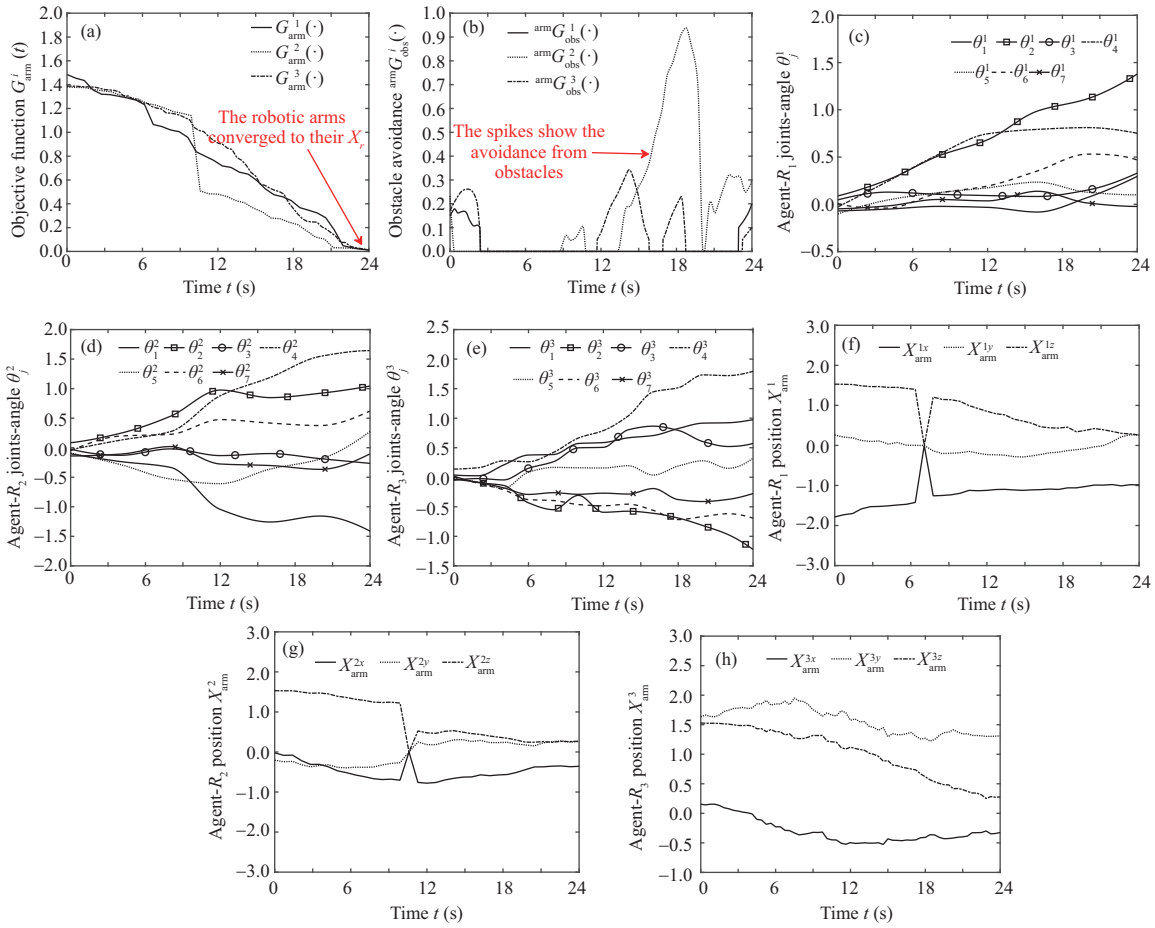



**Figure 4** (Color online) The "gripping and lifting" phase of the simulation. (a) shows the convergence of the objective function of all three agents. (b) shows the objective function values for the avoidance of the obstacles. (c)–(e) show the joints-angle of the agents and (f)–(h) show the positions of their end-effectors.

The complete optimization problem for this particular example includes (20), (34)–(36). The final optimization problem is given as

$$\gamma_1\left(\min_{\Phi^i_g}||\boldsymbol{X}^i_r - F(\boldsymbol{\Phi}^i_g(t))||_2\right) + \gamma_2\left(\min_{\boldsymbol{P}_{\text{mob}}}\text{dist}(\boldsymbol{X}^i_r, \boldsymbol{P}^i_{\text{mob}})\right) + \frac{\gamma_3}{\min_{\text{dist}}\ \text{dist}(\text{Obs}, \boldsymbol{P}^i_{\text{mob}}, V^i_j(\theta_j))} \tag{37}$$

$$\text{s.t.}\ \ \text{dist}(\text{Obs}, \boldsymbol{P}^i_{\text{mob}}) > r_b,\ \ g^i_{\text{mob}}(\dot{x}_b, \dot{y}_b, \dot{\phi}) = 0,\ \ \dot{v}^i_r < \dot{v}^i_{\max},\ \ \dot{v}^i_l < \dot{v}^i_{\max},$$

$$\theta^{i+}_j < \theta^i_j < \theta^{i-}_j,\ j \in \{1, 2, \ldots, M\},$$

$$Z(P_k) = c,\ k \in \{o, 1, 2, \ldots, K\},$$

$$||P_i - P_o|| = d_{io},\ i \in \{1, 2, \ldots, K\},$$

$$||P_1 - P_2|| = d_{12},\ \ ||P_1 - P_3|| = d_{13},\ \ ||P_2 - P_3|| = d_{23},$$

$$\gamma_1 + \gamma_2 + \gamma_3 = 1,\ \ 0 \leqslant \gamma_1, \gamma_2, \gamma_3 \leqslant 1,$$

where $j$ represents the total number of links in the robotic agent $i$. It is the more elaborative form of the general concept presented in (20), as it includes the real-world smart-home constraints. As mentioned earlier, there is no active communication between the robots; BASZNN works as a central control unit that sends and receives data from the robots. Based on the objective function value, it tunes them, such that they work cooperatively. The above-formulated problem includes all the constraints necessary to move the table around the room safely in a cooperative manner. The algorithm will accept only those objective function values that lie within these constraints, i.e., $H < H_{\text{best}}$, and discard the rest, as shown in Algorithm 1. In other words, BASZNN does not require a separate framework for cooperative planning, as the algorithm serves as a central-unit to control the coordination between agents.

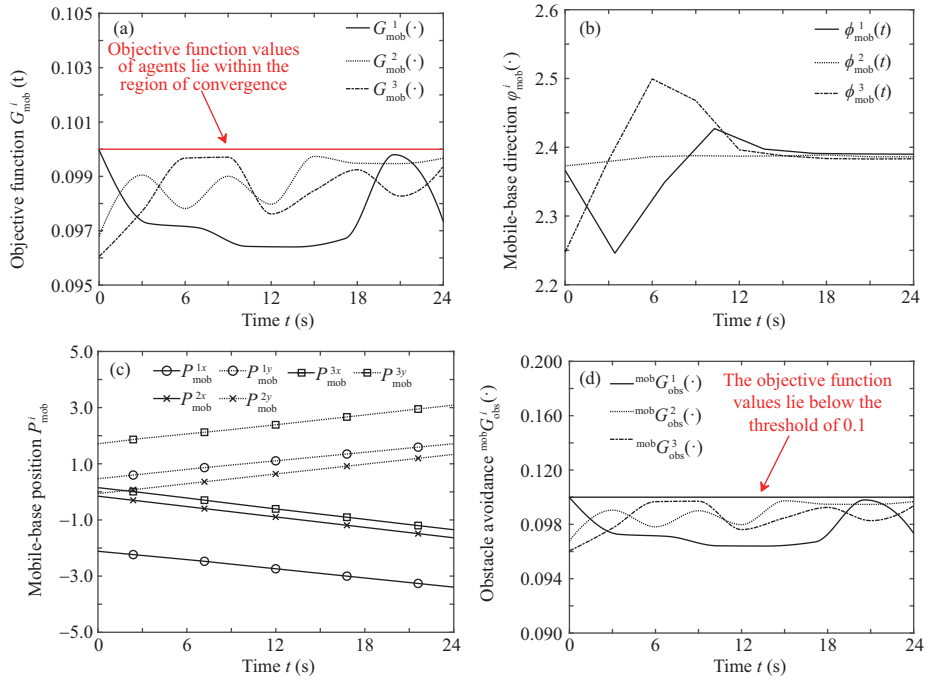



**Figure 5** (Color online) The "following" phase of the simulation. (a) shows the convergence of the objective function of all three agents. (b) shows the direction of agents while following the person $\boldsymbol{P}_o$; likewise, (c) shows their positions. Lastly, (d) shows the objective function values for the avoidance of the obstacles, which are below threshold 0.1, as there is no obstacle in their way.

The novelty of BASZNN is its adaptability to different problems without changing its framework. For example, Eq. (37) is an objective function $G_i(\cdot)$ from (25), which is also mentioned in Algorithm 1. The formulated problem in (37) is fed as an objective function $G_i(\cdot)$ to BASZNN, and the algorithm solves it iteratively until it converges to the optimal solution. Likewise, for the different scenarios, we have a different optimization problem. Still, for BASZNN, it is merely an objective function $G_i(\cdot)$, which is required to converge to its optimal solution by solving it iteratively.

### 4.3 Simulation results

For the environment of simulation, we used V-REP (robotic toolkit), and for the computation of BASZNN, we used MATLAB. The robotic agents are composed of Pioneer P3-DX (mobile-base), and KUKA LBR IIWA 7 (7-DOF robotic arm). The results obtained are divided into three phases: (1) approaching, (2) gripping and lifting, and (3) following. In the "approaching" phase, the assistive agents will approach the table. In the "gripping and lifting" phase, the agents will extend their arms to lift the table. Lastly, in the "following" phase, the agents will follow the coordinates of the person and move the table to the goal position. We set the BASZNN parameters as follows: $\lambda = 0.1$ (21), $\alpha = 10$ (26), $k = -10$ (29), and $\text{sgn} = [-1, 1]$ (29). We will discuss the results for each phase in detail.

The simulation results of the "approaching" phase are shown in Figure 3. BASZNN optimizes the weights for different phases of the task. For "approaching" phase the optimal weights turned out to be $\gamma_1 = 0.14, \gamma_2 = 0.57$, and $\gamma_3 = 0.27$. The sum of the optimal weights is 0.98 which almost obeys the equality constraint, i.e., 1. The robots are provided with the reference trajectory to reach the table (target) while avoiding the obstacles. The simulation in this phase lasted for around 16 s, and all three agents reached their respective target position in no time. Figure 3(a) shows the objective function value $G^i_{\text{mob}}(\cdot)$ of assistive agents. It can be seen that the robots reach their target position (near the table) as the objective function value monotonically decreases. It can also be observed that there are few spots for each agent where the value of $G^i_{\text{mob}}(\cdot)$ increases; that is because of the obstacle avoidance term in the objective function as the robot approaches an obstacle. Figure 3(b) shows the direction $\phi^i_{\text{mob}}$ of the agents. The direction of the first agent $R_1$ did not change much, whereas, there is an obvious change of the directions for $R_2$ and $R_3$. Figure 3(c) shows the positions of the agents, i.e., $\boldsymbol{P}^i_{\text{mob}} = [P^{ix}_{\text{mob}}, P^{iy}_{\text{mob}}]$. The table almost lies in the center of the room with coordinates $[0, 0]$, and we can see that the position

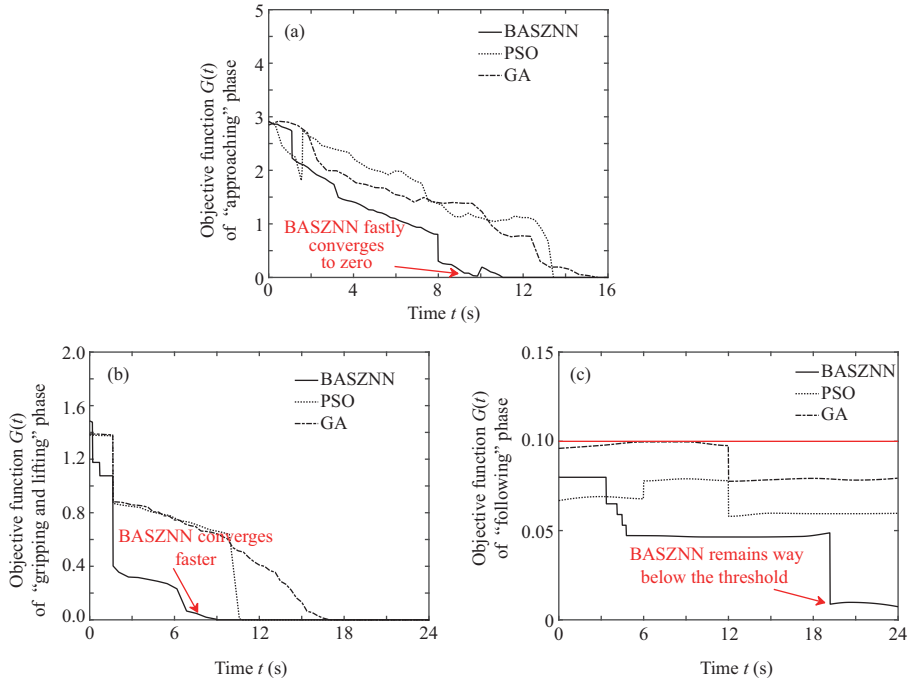



**Figure 6** (Color online) Comparison between BASZNN and two state-of-the-art known metaheuristic algorithms, i.e., PSO and GA. In all three phases, (a) approaching, (b) gripping and lifting, and (c) following, BASZNN outperforms the two algorithms as it has faster convergence towards the optimum solution.

**Table 1** Comparison between BASZNN, PSO, and GA

| | Approaching | | Gripping and lifting | | Following | |
|---|---|---|---|---|---|---|
| | Time ($t$) | $G_{\min}$ | Time ($t$) | $G_{\min}$ | Time ($t$) | $G_{\min}$ |
| BASZNN | 11.19 | $10^{-4}$ | 9.01 | $10^{-6}$ | 19.21 | $10^{-3}$ |
| PSO | 13.41 | $10^{-3}$ | 10.92 | $10^{-4}$ | 12.06 | 0.067 |
| GA | 15.34 | 0.01 | 17.04 | $10^{-3}$ | 12.07 | 0.077 |

of all three agents is approaching towards $[0, 0]$. Although we discussed the obstacle avoidance earlier, Figure 3(d) gives a better idea of how the agents avoided the obstacles in their way.

The simulation results of the "gripping and lifting" phase are shown in Figure 4. The optimal weights for the optimization problem (37) are as follows: $\gamma_1 = 0.48, \gamma_2 = 0.12$, and $\gamma_3 = 0.35$. The sum of the optimal weights 0.95 almost approaches the equality constraint, i.e., 1. The agents were provided with their respective grip points on the table. The $\gamma_1 = 2$ kept high to give more weightage to the manipulator arm, so that it reached out to grab the table (target). The simulation lasted for around 24 s. Figure 4(a) shows the objective function values $G^i_{\text{arm}}(\cdot)$, and it shows the same decreasing and fast convergence trend. Likewise, Figure 4(b) shows the obstacle avoidance, for robotic arms as the only obstacle was the table except for the gripping point $\boldsymbol{X}^i_r$. Figures 4(c)–(e) show the joints angle $\theta^i_j$, $j \in [1, 2, 3, \ldots, M]$ of the agents. Lastly, Figures 4(f)–(h) show the coordinates of the end-effector $\boldsymbol{X}^i_{\text{arm}} = [X^{ix}_{\text{arm}}, X^{iy}_{\text{arm}}, X^{iz}_{\text{arm}}]$ of the agents.

The simulation results of the "following" phase are shown in Figure 5. The optimal weights for (37) are as follows: $\gamma_1 = 0.38, \gamma_2 = 0.41$, and $\gamma_3 = 0.18$. The sum of the optimal weights 0.97 almost approaches the equality constraint, i.e., 1, and also remains within 0 and 1. The simulation lasted for 24 s. Here it is worth mentioning that the agents were provided the reference trajectory, such that they follow the person. In this phase, obstacle avoidance is an intricate and complex task under several constraints, so we have given more weights to $\gamma_1$ and $\gamma_2$. Figure 5(a) shows the objective function values $G^i_{\text{mob}}(\cdot)$ of the agents. As the three additional constraints (34)–(36) are rigorous to follow, the points on the trajectory they followed with respect to the person $\boldsymbol{P}_o$ were kept very close, which is why the $G^i_{\text{mob}}(\cdot)$ always remains less than the set threshold 0.1. Figure 5(b) shows the direction $\phi^i_{\text{mob}}$ of the agents. Figure 5(c) shows the position $\boldsymbol{P}^i_{\text{mob}}$ of agents, as we can observe that agents are moving away from the center $[0, 0]$, and towards the new position of the table. Figure 5(d) shows the obstacle avoidance of the agents under a

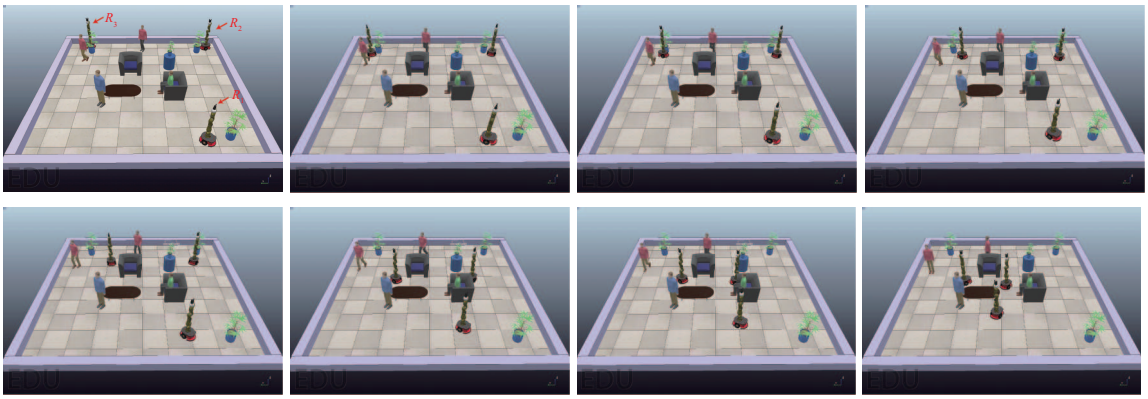


**Figure 7** (Color online) The "approaching" phase of the simulation where three agents (KUKA LBR IIWA 7 mounted on P3-DX) approached the table in a smart-home. All three agents are assigned three different positions at the corner of the table. It can also be seen that the agent $R_3$ successfully avoids both static and dynamic obstacles. Likewise, $R_2$ also succeed in planning its path through the obstacles.

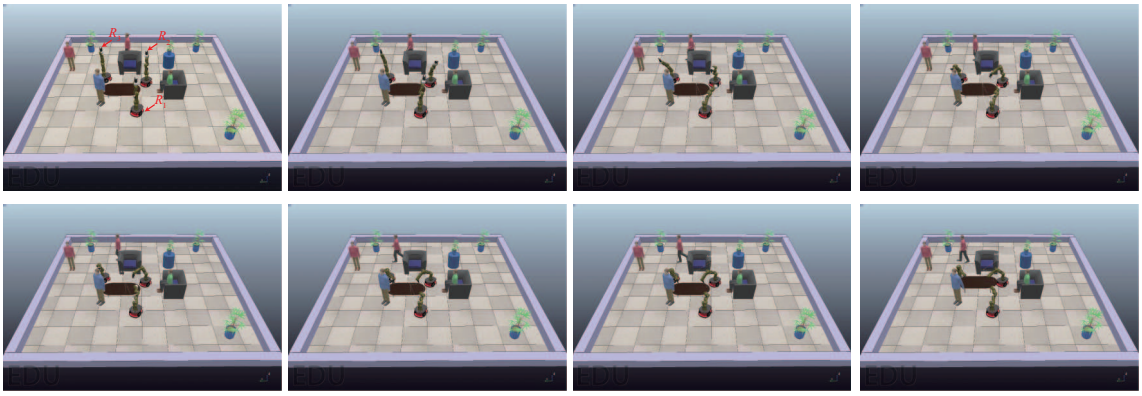


**Figure 8** (Color online) The "gripping and lifting" phase of the simulation where the arms of three agents (KUKA LBR IIWA 7 mounted on P3-DX) approached three corners of the table, first to grab it firmly and then to lift it under the constraint (34). The only obstacle for the agents was the table, except for the gripping point.

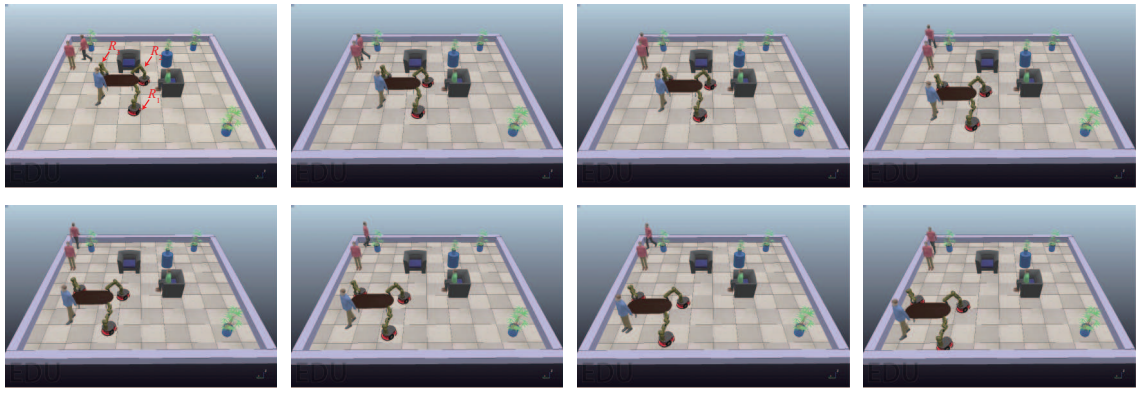


**Figure 9** (Color online) The "following" phase of the simulation where three agents (KUKA LBR IIWA 7 mounted on P3-DX) followed the person $\boldsymbol{P}_o$. Agents assisted the person to move the table from the center of the room to the corner just by following his coordinates. For the stability of the table, all the agents followed the constraints (34)–(36), successfully.

set threshold of 0.1, and shows almost no obstacle was in their way.

We also compared the performance of BASZNN with two known metaheuristic algorithms, i.e., PSO and GA. We performed all three phases of the task mentioned above with all three algorithms, and we computed the convergence of objective function values. Figures 6(a) and (b) show the "approaching" and "gripping and lifting" phases and more elaborated comparison is shown in Table 1. It can be seen that BASZNN converges to zero way faster than the other two algorithms. Likewise, Figure 6(c) shows the "following" phase. It can be seen that BASZNN remains way below the threshold of 0.1, as

compare to PSO and GA. The results show that despite the single particle nature of BASZNN, it still outsmarts the swarm algorithm. In future, we plan to extend the comparison between other state-of-the-art metaheuristic algorithms and BASZNN in different and more complex simulation environments to test the robustness of our algorithm. In the end, Figures 7–9 show the approaching, gripping and lifting, and following phases respectively of the simulation.

Here it is worth mentioning that the simulation is performed under a controlled environment. However, in a real-world scenario, the situation will be more complicated. One of our future work includes implementing the algorithm in a real-world environment, so that we can further analyze the performance of the algorithm and its time-space complexity. Despite the fast convergence of BASZNN, still, it has computational and time limitations. As mentioned earlier, it is polynomial in time and space; however, the complexity depends on the dimensions of the agents and the number of agents. Hence, when we increase the agents, both parameters increase as well, and it is difficult for a single particle to optimize them all. Another prospect includes the implementation of distributed BASZNN, where several BASZNN will work in parallel to accommodate the agents.

## 5 Conclusion

In this paper, we presented a framework for controlling the human-guided assistive agents in smart-homes. In the smart-home, household commodities sense the surroundings and act intelligently using the data available to them through cameras and sensors installed in them. Robotic agents play a vital role in smart-homes as they can work in aid for humans. However, the control of these agents is quite challenging in an environment full of static and dynamic obstacles. We formulated optimization problems for the control of robotic agents, including motion planning of mobile-base and redundant robotic arm, and avoidance of static and dynamic obstacles. We then unified them together and proposed a bio-inspired metaheuristic algorithm, BASZNN. The elementary BAS uses a concept of "virtual particles", which requires the computation of objective function three times, making it computationally expensive and time-consuming. Our proposed variant BASZNN overcomes this issue, and the distributing processing of ZNN makes it more efficient. To test the efficiency of our algorithm, we performed a simulation where three agents (KUKA LBR IIWA 7 mounted on P3-DX) assisted a person in moving a table around the room, and the BASZNN accomplished the task.

**Acknowledgements** This work was partially supported by CAAIXSJLJJ-2020-012A.

**References**

1 Cook D, Das S K. Smart Environments: Technology, Protocols, and Applications. Hoboken: John Wiley & Sons, 2004
2 Wilson G, Pereyda C, Raghunath N, et al. Robot-enabled support of daily activities in smart home environments. Cognitive Syst Res, 2019, 54: 258–272
3 Bevilacqua P, Frego M, Bertolazzi E, et al. Path planning maximising human comfort for assistive robots. In: Proceedings of 2016 IEEE Conference on Control Applications (CCA), 2016. 1421–1427
4 Machida E, Cao M, Murao T, et al. Human motion tracking of mobile robot with kinect 3D sensor. In: Proceedings of 2012 SICE Annual Conference (SICE), 2012. 2207–2211
5 Chung W, Kim H, Yoo Y, et al. The detection and following of human legs through inductive approaches for a mobile robot with a single laser range finder. IEEE Trans Ind Electron, 2012, 59: 3156–3166
6 Xu D, Zhao D B, Yi J Q, et al. Trajectory tracking control of omnidirectional wheeled mobile manipulators: robust neural network-based sliding mode approach. IEEE Trans Syst Man Cybern B, 2009, 39: 788–799
7 Fierro R, Lewis F L. Control of a nonholonomic mobile robot using neural networks. IEEE Trans Neural Netw, 1998, 9: 589–600
8 Oubbati M, Schanz M, Levi P. Kinematic and dynamic adaptive control of a nonholonomic mobile robot using a RNN. In: Proceedings of 2005 International Symposium on Computational Intelligence in Robotics and Automation, 2005. 27–33
9 Kanoun O, Lamiraux F, Wieber P B. Kinematic control of redundant manipulators: generalizing the task-priority framework to inequality task. IEEE Trans Robot, 2011, 27: 785–792
10 Wang J, Li Y, Zhao X. Inverse kinematics and control of a 7-DOF redundant manipulator based on the closed-loop algorithm. Int J Adv Robotic Syst, 2010, 7: 37
11 Ling S, Wang H, Liu P X. Adaptive fuzzy dynamic surface control of flexible-joint robot systems with input saturation. IEEE/CAA J Autom Sin, 2019, 6: 97–107
12 Wang H, Liu P X, Zhao X, et al. Adaptive fuzzy finite-time control of nonlinear systems with actuator faults. IEEE Trans Cybern, 2020, 50: 1786–1797
13 Wang H, Liu X P, Xie X, et al. Adaptive fuzzy asymptotical tracking control of nonlinear systems with unmodeled dynamics and quantized actuator. Inf Sci, 2021, 575: 779–792
14 Shang M, Luo X, Liu Z, et al. Randomized latent factor model for high-dimensional and sparse matrices from industrial applications. IEEE/CAA J Autom Sin, 2019, 6: 131–141
15 Luo X, Zhou M C, Xia Y N, et al. An efficient non-negative matrix-factorization-based approach to collaborative filtering for recommender systems. IEEE Trans Ind Inf, 2014, 10: 1273–1284

16 Luo X, Zhou M C, Li S, et al. A nonnegative latent factor model for large-scale sparse matrices in recommender systems via alternating direction method. IEEE Trans Neural Netw Learning Syst, 2016, 27: 579–592
17 Chen D, Zhang Y, Li S. Tracking control of robot manipulators with unknown models: a Jacobian-matrix-adaption method. IEEE Trans Ind Inf, 2018, 14: 3044–3053
18 Chyan G S, Ponnambalam S. Obstacle avoidance control of redundant robots using variants of particle swarm optimization. Robotics Comput-Integrated Manufact, 2012, 28: 147–153
19 Zhao J, Zhao L, Liu H. Motion planning of hyper-redundant manipulators based on ant colony optimization. In: Proceedings of 2016 IEEE International Conference on Robotics and Biomimetics (ROBIO), 2016. 1250–1255
20 Li S, Zhang Y, Jin L. Kinematic control of redundant manipulators using neural networks. IEEE Trans Neural Netw Learn Syst, 2017, 28: 2243–2254
21 Li S, Wang H, Rafique M U. A novel recurrent neural network for manipulator control with improved noise tolerance. IEEE Trans Neural Netw Learn Syst, 2018, 29: 1908–1918
22 Li S, Zhou M C, Luo X. Modified primal-dual neural networks for motion control of redundant manipulators with dynamic rejection of harmonic noises. IEEE Trans Neural Netw Learn Syst, 2018, 29: 4791–4801
23 Cheng L, Liu W, Yang C, et al. A neural-network-based controller for piezoelectric-actuated stick-slip devices. IEEE Trans Ind Electron, 2018, 65: 2598–2607
24 Yang C, Peng G, Li Y, et al. Neural networks enhanced adaptive admittance control of optimized robot-environment interaction. IEEE Trans Cybern, 2019, 49: 2568–2579
25 Yang C, Luo J, Liu C, et al. Haptics electromyography perception and learning enhanced intelligence for teleoperated robot. IEEE Trans Automat Sci Eng, 2019, 16: 1512–1521
26 Yang X S, Hosseini S S S, Gandomi A H. Firefly algorithm for solving non-convex economic dispatch problems with valve loading effect. Appl Soft Comput, 2012, 12: 1180–1186
27 Zhu Z, Zhang Z, Man W, et al. A new beetle antennae search algorithm for multi-objective energy management in microgrid. In: Proceedings of 2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA), 2018. 1599–1603
28 Khan A H, Li S, Luo X. Obstacle avoidance and tracking control of redundant robotic manipulator: an RNN-based metaheuristic approach. IEEE Trans Ind Inf, 2020, 16: 4670–4680
29 Khan A T, Cao X W, Li S, et al. Quantum beetle antennae search: a novel technique for the constrained portfolio optimization problem. Sci China Inf Sci, 2021, 64: 152204
30 Khan A T, Li S, Cao X. Control framework for cooperative robots in smart home using bio-inspired neural network. Measurement, 2021, 167: 108253
31 Sun Y, Zhang J, Li G, et al. Optimized neural network using beetle antennae search for predicting the unconfined compressive strength of jet grouting coalcretes. Int J Numer Anal Methods Geomech, 2019, 43: 801–813
32 Wu Q, Shen X, Jin Y, et al. Intelligent beetle antennae search for UAV sensing and avoidance of obstacles. Sensors, 2019, 19: 1758
33 Fei S W, He C X. Prediction of dissolved gases content in power transformer oil using BASA-based mixed kernel RVR model. Int J Green Energy, 2019, 16: 652–656
34 Li Q, Wei A, Zhang Z. Application of economic load distribution of power system based on BAS-PSO. In: Proceedings of IOP Conference Series: Materials Science and Engineering, 2019. 072056
35 Asif M, Khan M A, Abbas S, et al. Analysis of space & time complexity with PSO based synchronous MC-CDMA system. In: Proceedings of 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), 2019. 1–5
36 Ren S, Xie Y, Yang X, et al. A method for optimizing the base position of mobile painting manipulators. IEEE Trans Automat Sci Eng, 2017, 14: 370–375
37 Kim J E, Boulos G, Yackovich J, et al. Seamless integration of heterogeneous devices and access control in smart homes. In: Proceedings of 2012 8th International Conference on Intelligent Environments, 2012. 206–213
38 Cameron S. Enhancing GJK: computing minimum and penetration distances between convex polyhedra. In: Proceedings of International Conference on Robotics and Automation, 1997. 3112–3117
39 Jin L, Li S, Liao B, et al. Zeroing neural networks: a survey. Neurocomputing, 2017, 267: 597–604
40 Zhang Y, Li S, Xu B. Convergence analysis of beetle antennae search algorithm and its applications. 2019. ArXiv: 1904.02397