Auton Agent Multi-Agent Syst (2014) 28:72–100
DOI 10.1007/s10458-012-9213-x

Experimental studies on chemical concentration map building by a multi-robot system using bio-inspired algorithms

Mirbek Turduev · Gonçalo Cabrita · Murat Kırtay · Veysel Gazi · Lino Marques

Published online: 14 December 2012

Abstract In this article we describe implementations of various bio-inspired algorithms for obtaining the chemical gas concentration map of an environment filled with a contaminant. The experiments are performed using Khepera III and miniQ miniature mobile robots equipped with chemical gas sensors in an environment with ethanol gas. We implement and investigate the performance of decentralized and asynchronous particle swarm optimization (DAPSO), bacterial foraging optimization (BFO), and ant colony optimization (ACO) algorithms. Moreover, we implement sweeping (sequential search algorithm) as a base case for comparison with the implemented algorithms. During the experiments at each step the robots send their sensor readings and position data to a remote computer where the data is combined,

This work was supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) under Grant No. 106E122 and by the European Commission under the GUARDIANS Project (FP6 contract No. 045269).

M. Turduev
Department of Electrical and Electronics Engineering, TOBB University of Economics and Technology, Söğütözü Cad. No: 43, 06560 Ankara, Turkey
e-mail: mirbek.turduev@etu.edu.tr

G. Cabrita · L. Marques
Department of Electrical and Computer Engineering, Institute of Systems and Robotics, University of Coimbra, 3030-290 Coimbra, Portugal
e-mail: goncabrita@isr.uc.pt

L. Marques
e-mail: lino@isr.uc.pt

M. Kırtay
Department of Computer Science, Özyeğin University, Çekmeköy Campus, 34794 İstabul, Turkey
e-mail: murat.kirtay@ozu.edu.tr

V. Gazi (✉)
Department of Electrical and Electronics Engineering, Istanbul Kemerburgaz University, Mahmutbey Dilmenler Caddesi No: 26, 34217 Bağcılar, Istanbul, Turkey
e-mail: veysel.gazi@kemerburgaz.edu.tr

filtered, and interpolated to form the chemical concentration map of the environment. The robots also exchange this information among each other and cooperate in the DAPSO and ACO algorithms. The performance of the implemented algorithms is compared in terms of the quality of the maps obtained and success of locating the target gas sources.

1 Introduction

1.1 Robotic olfaction

In recent years the number of studies on application of robotic odor-sensing technology has increased substantially. Mobile robots equipped with chemical sensors can be useful for a number of application areas including safety, security, and environmental inspection. Instead of humans, robots can be dispatched to areas with odor contamination for inspection, or providing continuous monitoring of the contaminated environment for specific characterization of the odor. Employing multiple robots can further improve performance since they can do the search cooperatively in parallel. To achieve these goals there is a need for developing efficient algorithms for coordination between robots as well as robotic search, exploration, and coverage. For that purpose various approaches might be developed based on extremum seeking, gradient or hill climbing, multi-objective optimization algorithms, game theoretic strategies, negotiation based algorithms, classical methods such as sweeping, extending single-robot search, exploration, and coverage algorithms, and others. Another possibility is to use bio-inspired algorithms. Nature based algorithms such as particle swarm optimization (PSO), artificial immune systems, bacterial foraging optimization (BFO), and ant colony optimization (ACO) have been implemented in various applications including pollution mapping [1], mobile robot path planning [2], job scheduling [3], and traveling sales person problems [4]. In this paper, we employ bio-inspired algorithms to address the problem of determining the gas distribution in indoor environments by a group of mobile robots equipped with on-board gas sensors.

1.2 Problem definition

Consider an application in which a group of robots are required to perform a search in an unknown environment contaminated with a chemical substance. The chemical substance is possibly dangerous for human beings. Therefore, search by autonomous robots is more suitable to the problem. Moreover, employing multiple robots in parallel search may possibly lead to a faster and more effective performance. Assume that the robots are equipped with the necessary hardware and are able to sense the chemical (or the set of chemicals) which contaminate the environment. The objective is to build the map of chemical concentration as well as to determine the region with the highest concentration of the contaminant. Such information might be useful for experts who may determine whether the concentration level of the chemical is within tolerable levels or constitute a danger. If there are more than one chemicals contaminating the environment and the robots are equipped with chemical sensor arrays and can sense and distinguish different chemicals, it might be possible to determine the composition of the chemicals contaminating the environment. This, on the other hand, can provide extra useful information about the sources and severity of contamination. Higher chemical concentration areas usually occur very close to the sources of chemicals and determining the regions with high concentration might give information of the positions of the leaks (i.e., chemical sources) contaminating the environment. Such a situation can arise

in, for example, a building (such as a large warehouse) under fire where burning of certain chemicals might be dangerous since certain levels of some chemicals might lead to explosion. Moreover, the composition of the chemicals might provide information about the burning materials. All this information can be collected by the robots without endangering the lives of firefighters and critical decisions might be taken (e.g., a decision such as whether to go into the building or not) based on the information obtained by the robots. Other applications can include disaster situations in chemical or nuclear plants (provided that the robots are equipped with the necessary sensing hardware) and security or defence applications.

In order to be able to perform efficient search the robots should be able to pass their sensor readings as well as other needed information among each other. Moreover, they are required to pass the information they have gathered to a remote computer (a base station) where the data should be collected and combined and visual information must be provided in real-time to an operator. The task is to search an unknown environment with multiple-robots with the objective to locate the highest concentration of the gas and build a real-time map of the gas distribution.

1.3 Related work

Odor source localization and mobile robot olfaction have been investigated in various studies [5–9]. The studies include approaches such as swarming algorithms [5], hidden Markov methods [6], and Bayesian inference methods [7]. Similarly, integrating odor classification and gas distribution mapping [9], and utilizing Gaussian weighted functions and concentration gridmaps [8] for chemical gas concentration mapping of unknown environments have been also considered. The studies in [6–9] utilize a single agent, whereas the work in [5] considers multiple agents. Multiple odor sources are considered only in [9]. An environment with multiple contaminating sources is expected to be less ideal (more dynamic) and bring new challenges. Therefore, algorithms developed for static environments might not perform satisfactorily in applications involving multiple contaminating sources. Moreover, studies involving only simulation might overlook the practical/experimental difficulties which might be present in applications with real contaminating chemical sources. Starting from this point this article focuses on multi-robot bio-inspired algorithms for building chemical concentration map of an unknown indoor environment contaminated by a real chemical gas. In particular, we consider decentralized asynchronous particle swarm optimization (DAPSO), BFO, and ACO algorithms as high-level path planning and search strategies.

The PSO algorithm was introduced by Eberhart and Kennedy and used for optimization of continuous nonlinear functions [10]. In [11,12] Marques and his colleagues adapted it to the multi-robot odor searching problem, investigating both theoretically and experimentally the application of a PSO inspired search strategy for detecting odor sources across large spaces. There are also studies on adapting or utilizing PSO to multi-robot search problems [13–17]. The DAPSO algorithm implemented in this article has important differences with these algorithms and allows for dynamic neighborhood topology and time delays in information exchange in addition to totally decentralized and asynchronous operation.

The BFO algorithm was developed by Passino [18] inspired from chemotactic behavior of an E.*coli* bacterium while searching for nutrient rich environments [18,19]. It is another algorithm which has been adapted and implemented in multi-robot search applications to monitor plumed environments in a phototaxis experiment [20], and to determine pollution source location [1].

The ACO algorithm was developed by Dorigo [4] inspired by the foraging behavior of ant colonies. Up until now, the algorithm has been successfully implemented on various problems

such as traveling sales person, machine learning, job scheduling, and path planning [4]. It has been also utilized for odor source localization [21,22] in a Matlab simulation environment. In this article we use ACO to build chemical gas concentration map through mobile robots with real chemical gas.

Note that the work here has important differences from related works presented above. First of all, we use a real robotic system which operates in a real chemical (ethanol gas) environment. The works in the literature are mostly simulation based and/or either use smooth non-experimental data or operate on a previously collected and smoothed data. Moreover, substantial amount of the studies are performed using a single robot. Furthermore, most of the studies are on source localization. Since our robots perform search based on sensor reading information they are equipped with chemical sensing hardware. Most of the related studies which use simulated or other data just assume that the robots have sensing capability. Therefore, the works using pre-collected and smoothed or simulated data are not exposed to measuring imperfections and other effects to the same extend as our work here. Moreover, in addition to locating the areas with high gas concentration (and therefore the contamination sources) here we also perform real-time chemical concentration map building. In other words, the obtained gas concentration is visualized in real-time on a remote computer (a base station) in a realistic scenario. In fact, building the map of the chemical gas is the primary task in this study.

In addition to the primary considered bio-inspired algorithms (i.e., DAPSO, BFO, and ACO) we have implemented also sweeping (sequential search algorithm) and Canonical PSO (CPSO) algorithms for performance comparison purposes. This study constitutes an extended and comparative version of the studies in [23–25].

2 Experimental setup

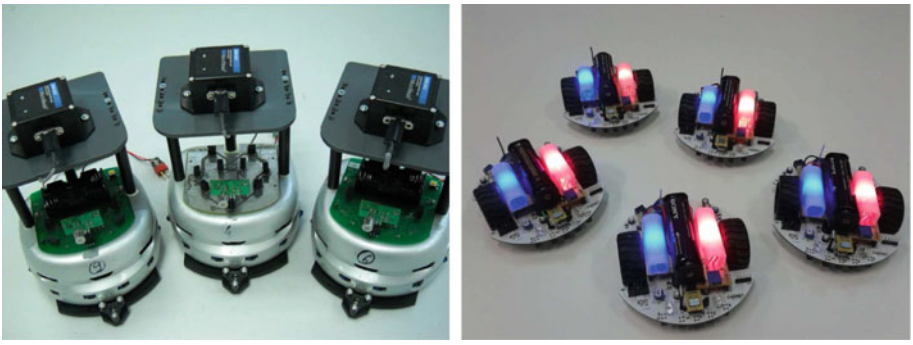
2.1 Environment and robots

Implementations were performed in two experimental setups available at our laboratories. Note that one of these setups was also used in [23–25]. Therefore, part of the description here follows along the same lines as the description in [23–25]. Initial set of experiments were performed with Khepera III mobile robots equipped with the "kheNose" sensing system [26] and Figaro TGS2620 alcohol sensors. The second set of experiments were performed with miniQ mobile robots equipped with e2v MiSC5524 gas sensors. The experimental setups used are shown in Fig. 1. The first setup consists of $3.40 \times 2.40 \times 1.35$ m dimensional experimental arena covered by a *transparent vinyl cover* (Fig. 1a). In order to provide gas circulation the front right corner of the covered arena is left open to let air flow inside it. Similarly, the second setup consists of completely enclosed 3.0×4.0 m arena delimited by four walls where two of them are made of an honeycomb-like pattern to allow for the air to flow between the two 3.0 m opposing walls. It includes an array of controllable fans that allows generating different patterns of air flow. These setups constitute small scale representations of a large building (such as a warehouse) filled with a contaminating chemical gas. Ethanol, which is a volatile and colorless liquid, is used as chemical gas. For that purpose we use an air bubbler system composed of bottles half filled with ethanol, an air pump, and plastic tubes to inject the evaporated alcohol into the enclosed environment. Odor sources are located at the ceiling of the covered area and inject the gas downward.

In the implementations performed in Setup I differentially driven Khepera III robots shown on Fig. 2a are used. They are equipped with KoreBotLE extension cards which have Intel

(a) Setup I **(b)** Setup II

Fig. 1 Enclosed experimental setups



(a) KheperaIII robots **(b)** MiniQ robots

Fig. 2 Robots used in the implementations

PXA255 processors working at 400 MHz. In order to determine/measure sensor readings a DSPIC30F5011 microcontroller working at 60 MHz is used and the sensors communicate with the microcontroller via the I^2C bus. Each wheel motor is controlled by its own PID controller implemented on a PIC18F4431 microcontroller which is also used for measuring the odometry information of the robot. The robot localization is improved by a MicroStrain 3DM-GX2 Inertial Measurement Unit (IMU), shown on top of the robots in Fig. 2a. This IMU combines a high-performance triaxial gyro with a triaxial accelerometer, and a triaxial magnetometer to provide complimentary displacement and orientation estimates. Orientation errors are dominant in odometry localization system errors. The employed IMU devices allow reducing these errors. Simple thresholding method is used to integrate the data from the IMUs to the odometry data. Basically, when the change in robot orientation is larger than a predefined threshold, the IMU data is used for calculating the turn rate and therefore for localization, otherwise the odometry data is used for localization. More sophisticated localization algorithms are also possible. However, it should be noted that localization is not the main emphasis of this article. Moreover, experimental tests using the UMBmark [27,28] procedure performed using a $50 \times 50\,\text{cm}^2$ path show that in the considered experimental arena the self-localization error for the Khepera III robots using the above odometry correction is small. In particular, it was shown in [29] that while the mean errors for normal odometry over 20 experiments are $\varepsilon_x = 3.11\,\text{cm}, \varepsilon_y = 3.33$ cm, and $\varepsilon_\theta = 7.86\,^\circ$, which denote the

average return position and orientation errors, the same for the IMU corrected odometry are $\varepsilon_x = 0.73\,\text{cm}, \varepsilon_y = 0.71\,\text{cm}$, and $\varepsilon_\theta = 1.55\,^\circ$.

In the implementations performed in Setup II differentially driven MiniQ robots shown on Fig. 2b are used. MiniQs are small and cheap robots based on the Arduino micro-controller powered by an Atmega328 running at 16MHz. They were modified to achieve olfactory swarming and were equipped with an e2v MiSC5524 gas sensors. They were also equipped with two LEDs (one red and one blue) for usage with an Arecont MegaVideo IP camera for correcting the odometry of the robots and to provide global localization. Due to the low processing power of the miniQ an XBee module is used for communication with a computer. The robots are in charge of calculating their odometry and receiving linear and angular velocity commands. The computer runs the navigation, odometry correction, global localization and the swarming algorithms. Odometry correction and global localization is achieved using SwisTrack, a software designed for tracking robots. In this case the dual tag composed by the red and blue LEDs allows to acquire the pose of the robot (position and orientation).

We would like to emphasize once more that the emphasis here is on the cooperative chemical concentration map building by multiple robots. The localization we use is sufficient for the laboratory experiments in this article. For applications in environments such as large warehouse the robots should have also reliable localization capability.

2.2 Chemical sensors

KheNose is a modular olfactory sensing system for Khepera III mobile robots. This device is composed by a main board, with robot interfacing and processing capabilities, and an array of gas sensing nostrils, a temperature and humidity sensor, and up to three small thermal anemometer boards.[1] Each sensing board contains a Transducer Electronic Data Sheet stored in an EEPROM memory, providing plug-and-play capabilities to the system [26]. The main board contains a Microchip dsPIC33F controller which acquires all the analog and digital information from the sensors, processes that data and sends it to the Khepera III KoreBotLE extension board through an I^2C interface. The system can operate as an odor compass, being able to measure airflow intensity and direction while it classifies the detected odors [30]. The odor classification is achieved by means of a feedforward neural network. This classification can be done using the steady-state response of the gas sensor array or, for faster classification, using the coefficients of the discrete wavelet transform of the sensor array transient response [31].

As mentioned above, the minQs were equipped with e2v MiSC5524 gas sensors. This choice is related to the fact that the miniQ has low torque motors, which combined with the low resolution from the single channel encoders makes it difficult to move the robot at low speeds. The MiSC5524 from e2v is faster than the Figaro TGS2620 and is thus more appropriate for the miniQ.

2.2.1 Calibration of multiple olfactory systems

The conductance G of tin oxide gas sensors varies with the concentration C of a target reducing gas accordingly with following relationship [32]:

$$G = G_1 P_R{}^n \tag{1}$$

[1] In the current work, only the information provided by a Figaro TGS2620 metal oxide gas sensor was employed.

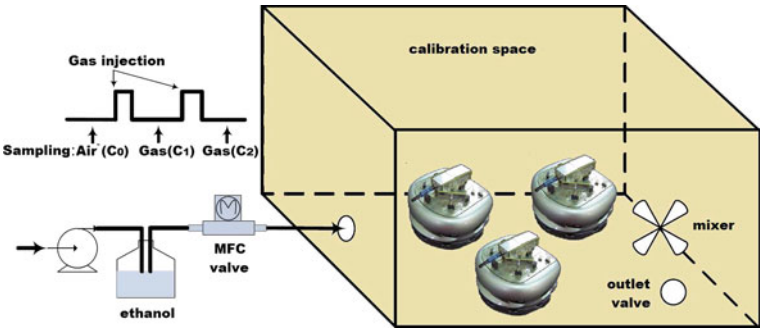


Fig. 3 Calibration setup

where G_1 is the conductance for a small concentration C_1 of the reducing gas, $P_R = C/C_1$ is the relative concentration of the gas, and n is a constant dependent from the gas and from the sensor.

In multiple robot olfactory experiments it is fundamental to have the sensing systems calibrated against the same standard values, so the measurements can be merged in a single concentration map. In the current experiments the following fast calibration method was implemented:

(1) All the employed robots/sensors were placed inside an enclosed acrylic box with $30 \times 30 \times 20\,\text{cm}^3$. The box contains a 5 cm diameter ventilator fixed on the top and a reference sensing system fixed in a lateral wall, containing a gas sensor and temperature and humidity sensors (Sensirion SHT21). The box contains an inlet through which fixed amounts of a target gas (ethanol vapor) can be inserted. The concentration inside the calibration space is kept homogeneous by the action of the mixing ventilator.
(2) The conductance in clean air was registered as G_{air}.
(3) A small volume of ethanol vapor was inserted into the calibration space and, after stabilization of the sensors, the conductance corresponding to the existing atmosphere concentration C_1 was registered as G_1.
(4) The same volume of ethanol was inserted into the calibration space and the conductance G_2 corresponding to the concentration $C_2 = 2C_1$ was registered.

Known amounts of ethanol vapor can be inserted into the calibration space using a large syringe or a mass flow controller, as shown in Fig. 3. In that figure the robots are inside an acrylic glass calibration box, containing a small fan mixer to homogenize the atmosphere.

After the above calibration procedure, the constant n of each sensor could be determined. In operation, for concentrations above C_1, Eq. (1) was employed and for concentrations below that value, a linear interpolation between the output to clean air and the output to C_1 was employed.

3 Potential functions and robot control

In order to navigate and also avoid robot to robot collisions artificial potential functions are used for low-level control of the robots. With this objective in order to move the robots (both Kheperas or miniQs) to their next way-points we use quadratic attractive potential functions and require the robots to move along their negative gradients. Also, in order to

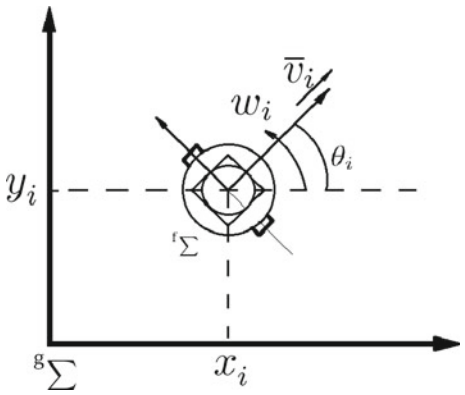


Fig. 4 Non-holonomic robot model

avoid collisions between the robots we use repulsive potential functions which are activated when the distances between robots become smaller than a predefined constant value D. In the Kheperas the repulsive potential forces are calculated using the readings of the infrared sensors. Since the miniQs do not have proximity sensors collision avoidance is not employed in the experiments with them.

The motion of the robots in $\mathbb{R}^2$ can be represented with the unicycle agent model

$$\begin{aligned} \dot{x}_i(t) &= \bar{v}_i(t)\cos(\theta_i(t)), \\ \dot{y}_i(t) &= \bar{v}_i(t)\sin(\theta_i(t)), \\ \dot{\theta}_i(t) &= w_i(t) \end{aligned} \tag{2}$$

where $p_i(t) = [x_i(t), y_i(t)]$ is the position vector and $\theta_i(t)$ is the steering angle of agent i at time t. Its control inputs are the linear speed $\bar{v}_i(t)$ and the angular speed $w_i(t)$. The illustration of the non-holonomic robot model is shown in Fig. 4. Note that here the robots correspond to particles in the DAPSO and CPSO algorithms, to the bacteria in the BFO algorithm, and to the ants in the ACO algorithm. These bio-inspired algorithms are used for high-level path planning of the robots or basically for determining the way-points the robots should visit. In other words, at time t_k^i (iteration k for robot/agent i) corresponding to the kth way point $p_i(t_k^i) = [x_i(t_k^i), y_i(t_k^i)] \in \mathbb{R}^2$ in Cartesian coordinates robot i computes its (desired) next way point $p_i(t_{k+1}^i)$ using one of the mentioned bio-inspired algorithms. In order to move the robot from the kth way point $p_i(t_k^i)$ to the $(k+1)$th way point $p_i(t_{k+1}^i)$ we use artificial potential functions for low level control. In particular, we use the straight path between the two points and force the robot to move along the vector $\bar{p}(t_k^i) = p_i(t_k^i) - p_i(t_{k+1}^i)$ to reach $p_i(t_{k+1}^i)$. For that purpose we use a quadratic potential of the form

$$J_{attr}^i(t) = \frac{a}{2}\big(p_i(t) - p_i(t_{k+1}^i)\big)^2$$

where the constant parameter $a > 0$ is an attraction coefficient, $p_i(t)$ represents the current position of the robot at time $t \in [t_k^i, t_{k+1}^i)$, and $p_i(t_{k+1}^i)$ is the next way-point of the robot in the search space. Let us denote the negative gradient of the attractive potential $J_{attr}^i(t)$ with $G_{attr}^i(t)$ (i.e., $G_{attr}^i(t) = -\nabla_{p_i} J_{attr}^i(t)$). Then, we have

$$G_{attr}^i(t) = -a\big(p_i(t) - p_i(t_{k+1}^i)\big)$$

for $t \in [t_k^i, t_{k+1}^i)$. Note that with this configuration there is a linear attraction for each robot towards its respective way-point in the search space. In contrast, to avoid robot to robot collisions a nonlinear (and unbounded) repulsion potential is used in the form

$$J_{rep}^{ij}(t) = b\left(\ln(\|\bar{p}_{ij}(t)\|) + \frac{D}{\|\bar{p}_{ij}(t)\|}\right)$$

where $\bar{p}_{ij}(t) = \big(p_i(t) - p_j(t)\big)$ and the constant parameter $b > 0$ is a repulsion coefficient. This potential, which is used in the experiments with the Khepera robots, is activated only in case there is robot j in the vicinity of robot i at a distance smaller than a predefined distance D. Calculating its gradient $G_{rep}^{ij}(t)$ at time $t \in [t_k^i, t_{k+1}^i)$ one obtains

$$G_{rep}^{ij}(t) = \begin{cases} b\bar{p}_{ij}(t)\left(\frac{1}{\|\bar{p}_{ij}(t)\|^2} - \frac{D}{\|\bar{p}_{ij}(t)\|^3}\right), & \|\bar{p}_{ij}(t)\| \le D \\ 0, & \|\bar{p}_{ij}(t)\| > D \end{cases}$$

which constitutes the artificial repulsive force to be inserted on robot i due to robot j in its vicinity.

Using the above, the total potential (field) $G^i(t)$ for robot i is determined as

$$G^i(t) = G_{attr}^i(t) + \sum_{j=1, j\neq i}^{N} G_{rep}^{ij}(t) \tag{3}$$

where N is the number of robots. Note also that although the summation of the repulsion terms is over all the other robots, only those which are within the detection range of robot i actually effect the value of $G^i(t)$. We use the local coordinate system of the robot for calculating the attraction and repulsion forces.

Since the robots are non-holonomic (i.e., they cannot move in the direction of the axis connecting the two main wheels) and since their orientations initially may not necessarily be along the vector potential, we define the desired direction of motion of agent i as

$$\theta_d^i(t) = \text{atan2}\Big(G_y^i(t), G_x^i(t)\Big), t \in [t_k^i, t_{k+1}^i),$$

where $G_x^i(t)$ and $G_y^i(t)$ represent the x and y components of the potential field $G^i(t)$, respectively. Then, for the orientation dynamics of the robot we use the simple proportional controller

$$w_i(t) = -\alpha\Big(mod((\theta_i(t) - \theta_{id}(t)) + \pi, 2\pi) - \pi\Big), t \in [t_k^i, t_{k+1}^i), \tag{4}$$

where $\theta_i(t)$ represents the current orientation of the robot at time t and $\alpha > 0$ is the proportional gain. This shifted mod operation in (4) results in the fact that the robot turns towards the desired orientation in the direction of the smaller angle between it's currents orientation and desired orientation vectors [33]. For the linear speed controller we use

$$\bar{v}_i(t) = \min\{\|G^i(t)\|, v_{max}\}, t \in [t_k^i, t_{k+1}^i), \tag{5}$$

where v_{max} is an upper bound on the linear speed of the robot.

In addition, because of the noise in the measurements of the infrared sensors, we augment the potential functions based collision avoidance of the Kheperas with a priority based robot to robot collision avoidance. In other words, when two robots get in a close collision distance the robot with smaller ID waits until the robot with larger ID avoids the collision and continues on its path.

In an environment with obstacles the above algorithm can easily be modified to incorporate potential functions based obstacle avoidance similar to robot to robot avoidance, using the information provided by the range sensors (assuming that the robots possess such sensors). Moreover, other low-level control algorithms can also be used instead of the potential functions method described here. The tasks of the low-level control are to move the robot from its current position to its next way point generated by its high-level path planner (i.e., DAPSO, BFO or ACO in this article) and to avoid collisions with other robots and (possibly) obstacles. Any algorithm which provides these capabilities can serve as a low-level controller. One issue to be considered in environment with obstacles is the situation which raises when a generated way-point occurs to be located in a location occupied by an obstacle. The robots must possess capability to detect such situations and to replan their next way-points.

4 Bio-inspired search algorithms

4.1 Particle swarm optimization

In this section we briefly describe the PSO algorithm which is one of the algorithms used as high-level path planning strategy. In particular, first we describe the standard or canonical PSO algorithm following which we discuss also the DAPSO algorithm used in this article. PSO is a population based direct search algorithm which is in general suitable for multi-robot search applications. However, there are also important differences which are discussed in [13]. Two of the main differences can be stated as: (i) the robots cannot jump to their next positions, whereas the particles can do; (ii) the communication range of the robots can be limited, whereas there is no such constraint for the particles. (See [13] for more details.) These differences can lead to degradation of performance or even failure of the PSO algorithm when it is applied directly to multi-robot search without any modification. We discuss the differences in detail in the subsection on the DAPSO algorithm. However, for completeness let us first discuss the synchronous basic PSO algorithm.

4.1.1 Canonical PSO (CPSO) based search algorithm

In this article we use the PSO version proposed by Clerc and Kennedy in [34] to which we refer to as Canonical PSO or CPSO for short. It uses a "constriction coefficient" to prevent the "explosion behavior" of the particles (see [34] for more details). At the kth iteration the update of particle i, $i = 1, \ldots, N$, can be described as

$$\begin{aligned} v_i(t_{k+1}^i) &= \chi\left[v_i(t_k^i) + \varphi_1^i(t_k^i)\Big(b_i(t_k^i) - p_i(t_k^i)\Big) + \varphi_2^i(t_k^i)\Big(g_i(t_k^i) - p_i(t_k^i)\Big)\right] \\ p_i(t_{k+1}^i) &= p_i(t_k^i) + v_i(t_{k+1}^i) \end{aligned} \tag{6}$$

where t_k^i is the update time. Note that in this article each robot is considered as a particle from PSO view-point. The robot dynamics operates in continues time t and the instances at which robot/particle i performs its kth iteration is denoted with t_k^i. Here $p_i(t_k^i) \in \mathbb{R}^2$ represents the position (way point) of the i'th particle at time t_k^i, $b_i(t_k^i) \in \mathbb{R}^2$ represents the best position of the i'th particle from time $t = 0$ to time $t = t_k^i$, $g_i(t_k^i) \in \mathbb{R}^2$ represents the best position of the (possibly time varying) neighborhood of the i'th particle from time $t = 0$ to time $t = t_k^i$. The value $p_i(t_{k+1}^i) \in \mathbb{R}^2$ which is calculated in (6) is the next (desired) way point to which the robot should move. The learning coefficients $\varphi_1^i(t_k^i) \in [0, \bar{\varphi}_1]^2$ and

$\varphi_2^i(t_k^i) \in [0, \bar{\varphi}_2]^2$ are two dimensional uniform random vectors. The constant parameter $\chi > 0$ is the constriction parameter which prevents the explosion behavior, i.e., particles having high velocities leading to their scattering in the search space. For efficient performance and prevention of the explosion behavior in (6) we use the components of the learning coefficient vectors $\varphi_1^i(t_k^i)$ and $\varphi_2^i(t_k^i)$ proposed by [34] as

$$\varphi_{1j}^i(t_k^i), \varphi_{2j}^i(t_k^i) \in [0, \bar{\varphi}_j], \; j = 1, 2; \; i = 1, \ldots, N. \tag{7}$$

The constriction parameter $\chi > 0$ is calculated with the relation (refer to [34])

$$\chi = \begin{cases} \frac{2\kappa}{\varphi - 2 + \sqrt{\varphi^2 - 4\varphi}}, & \text{if } \varphi > 4, \\ \kappa, & \text{else.} \end{cases} \tag{8}$$

Here $\varphi = \bar{\varphi}_1 + \bar{\varphi}_2$ and $\kappa \in [0, 1]$. Similar to the work in [35], considering $\bar{\varphi}_1 = \bar{\varphi}_2 = 2.05$ and $\kappa = 1$, the constriction parameter is calculated as 0.7298 for this implementation.

4.1.2 Decentralized asynchronous PSO (DAPSO) based search algorithm

As mentioned before there are important differences between robotic search and PSO search. The robots have to traverse the entire path between the current way-point and the next way-point calculated by the PSO algorithm. Since the distance between the current positions and the next way-points can be different for different robots in the CPSO a robot which arrives at its way-point earlier than the other robots has to wait for them in order to perform its iteration. Moreover, since the communication range of the robots is usually limited and the area to be searched can be large during the search process the robots may exit the communication range of their neighbors, which may result in indefinite wait and stall by the system. Moreover, permanent communication and robot failures can lead to failure of the overall search. Realizing these potential problems a DAPSO inspired multi-robot search algorithm which allows also for dynamic neighborhood and possible time delays is proposed

Algorithm 1 Pseudocode of the DAPSO Algorithm

```
Initialize the variables
Assign the first way points intentionally far from each other
Move (first way-point)
Assign the second way point randomly
while (Stopping criteria is not satisfied) do
  while (Agent has not arrived to its way point) do
    Move (next way-point)
    Update S(b_i)
    Listen to data from other robots
  end while
  Broadcast own S(b_i) information
  if (S(b_i_other) > S(g_i) or S(b_i) > S(g_i)) then
    Update S(g_i)
  else
    Use previous S(g_i)
  end if
  Calculate a new way point using Equation (6)
end while
```

Algorithm 2 Pseudocode of the **Move/Swim** Task/Behavior

```
Move/Swim(way-point)
while (Agent has not arrived to its way point) do
  Move toward the assigned way-point
  Read concentration data from the environment
  Send data to base station
  if (Collision distance to other robots) then
    Apply collision avoidance
  end if
end while
```

in [35] (which is also inspired by the earlier works on DAPSO in [36,37]). The realization of this algorithm on the robotic search problem here can be briefly described by the pseudocode given in Algorithm 1 (which is taken from [35] with slight modification) and illustrated in Fig. 5. In Algorithm 1 $S(b_i)$ refers to sensor reading at the best position of robot and $S(g_i)$ is the sensor reading at the global best position. Note once more that before initiating the algorithm the robots move to points away from each other for better coverage of the area. During this step also they continuously sense the environment and send the information to the base station. This behavior is used also in the BFO and ACO algorithms discussed below. The move behavior in Algorithm 1 is illustrated in Algorithm 2. It is also a generic behavior which is used in the algorithms below as well. One can see from the pseudocode in Algorithm 1 (which describes the operation of a single agent) that the agents operate in a decentralized fashion and there is no attempt to synchronize their operations. Moreover, they utilize only the data obtained without waiting for all other robots to finish their operations. A mathematical model for the DAPSO algorithm can be found in [36,37]. These properties turn advantageous in multi-robot applications over the CPSO algorithm. In particular, they solve the above mentioned shortcomings (i.e., waiting for exchange of information and potential temporary or permanent stall and failure to complete the task) of the CPSO algorithm. Moreover, the CPSO algorithm can suffer from scalability problems since every robot needs to communicate and exchange information with every other robot. In contrast, in the DAPSO algorithm the robots use only the information they have access to. The robot communication is in principle unidirectional and they just broadcast their information. The robots which are close enough to the broadcasting robot and receive its data use it in their DAPSO iteration. They do not even have to know how many robots there are in the group or in their proximity. In a large warehouse with dimensions much larger than the communication range of the robots, in the DAPSO algorithm the group may separate to disconnected subgroups operating in different regions of the environment without disrupting operation and each subgroup can continue its operation and complete its task in its region. Taking these issues into account one can state that compared to the CPSO algorithm the DAPSO algorithm will better scale to larger number of robots and to larger environments (relative to the communication range of the robots).

One can notice that there are three main tasks which the robots perform. These tasks are presented in Table 1. The "move" task is in principle the same for all algorithms considered in this article. The "acquire, process, and exchange information" task has commonalities and differences. The differences are mainly in the strategy/algorithms to acquire and process information. Moreover, not all algorithms require exchange of information and cooperation, although the data is sent to the base station in all cases. These differences are also reflected to the "explore and exploit" task which is the main path planning task used to determine

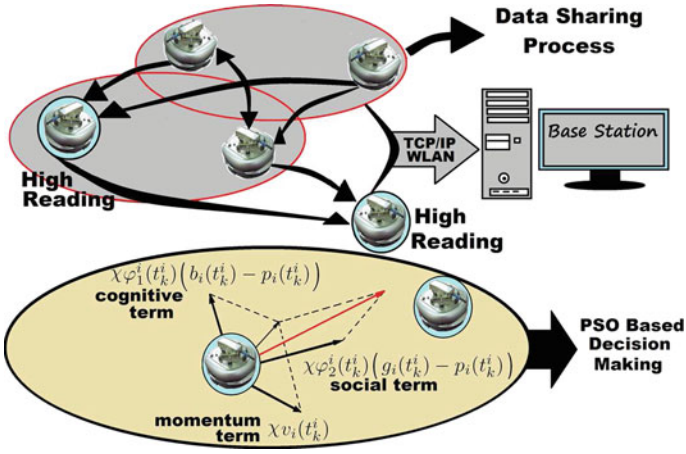


Fig. 5 Illustration of the PSO algorithm on mobile robots

Table 1 Tasks

Task	Explanation
Task 1: Explore and Exploit	Determine path for exploration.
	Assign way-points.
Task 2: Move	Move to target way-point.
	Motion control (with obstacle avoidance).
Task 3: Acquire, process, and exchange information	Read chemical concentration data, perform necessary calculations, send the data to the other robots and the base station.

the positions to visit. In particular, every algorithm has its own logic for calculating the next way-point the robot should visit for effective exploration.

4.2 Bacterial foraging optimization

4.2.1 General specifications of BFO

In this section we describe the original BFO algorithm proposed in [18]. BFO is one of the nature inspired algorithms which tends to imitate basic foraging behavior of bacteria. In that, bacterial motile behaviors, namely taxes, is triggered by the varying concentration of either nutrients or noxious substances in the environment. Bacteria movements can be categorized into two main motile behaviors which are tumbling and swimming (or running) which are achieved based on the clockwise or counterclockwise rotation of the flagella. Chemical substances in the environment enable bacteria to decide whether to swim or to tumble. The motion of bacteria can be described in terms of run intervals during which the cell swims approximately in straight line alternated with tumbles, when the organism undergoes a random reorientation.

Inspired from this behavior in the BFO algorithm the chemotactic behavior of the bacteria is represented by two activities which are namely a tumble and a swim. During the

jth chemotactic step, the position displacement of the ith bacterium can be represented as [18]

$$\theta_i(j+1,r,l) = \theta_i(j,r,l) + C(i)\phi_i(j) \tag{9}$$

where $C(i) > 0,\ i = 1,\ldots,N$ is basic chemotactic step size that defines the length of steps during runs; $\theta_i(j,r,l)$ indicates the position of the ith bacterium at jth chemotactic step in the rth reproductive loop of the lth elimination and dispersion event and $\phi_i(j)$ is a unit length random direction vector in the jth chemotactic step which is generated with orientation within the range $[0, 2\pi]$ in $\mathbb{R}^2$. In other words, $C(i)$ is the size of the step taken in random biased direction specified by a tumble in each chemotactic step. The bacterium compares the concentration/cost/fitness value at the position $\theta_i(j+1,r,l)$ and the concentration/cost/fitness value at the position $\theta_i(j,r,l)$ and makes a decision of the next chemotactic step. If the concentration value at the position $\theta_i(j+1,r,l)$ is better than the concentration value at the position $\theta_i(j,r,l)$, then the bacterium does not perform a tumble and swims $C(i)$ steps in the previously chosen direction $\phi_i(j)$ (i.e., the direction determined at the previous tumble). The bacterium continues its swims in the same direction as long as the new measured values are better than the old values or until a predefined maximum number N_s of swim steps is reached. In contrast, if the cost value at the position $\theta_i(j+1,r,l)$ is worse than the cost value at the position $\theta_i(j,r,l)$ then the bacterium will perform a tumble and a new random unit direction vector $\phi_i(j)$ is generated.

Execution of N_c chemotactic steps, leads the BFO algorithm to a reproduction step. In this step the algorithm sorts all bacteria according to their own accumulated cost, which depends on the values of the function being optimized (i.e., the concentration/cost/fitness) at the positions which the bacteria has visited so far, to distinguish the healthy bacteria which are the first half of the population size. Then, the bacteria which are located in the first half reproduce (i.e. make their own copy) and the bacteria in the second half are eliminated to protect the initial population size.

In nature, external dynamics, unpredictable environmental changes and animals, may lead to elimination/death of bacteria or apply irresistible force to bacteria colony to move from one place to another. To simulate such events on a bacteria colony, Passino introduced elimination-dispersal events N_{ed} in the BFO algorithm which should be repeatedly executed after a certain number of reproduction steps. Such events help the algorithm better explore the space and aid motion toward the global optimum. Moreover, to facilitate swarming and cooperation Passino introduced also the idea of cell-to-cell attraction. However, we do not consider these effects here. More information about the BFO algorithm can be found in [18]. Information about the behavior of the E.*coli* bacteria (the behavior of which the BFO algorithm is inspired from) can be found in [19].

4.2.2 Implementation of BFO on multi-robot search

In this section we describe the modified BFO algorithm implemented on the robots. It has some differences from the BFO algorithm developed in [18] and briefly described above. Similar to the case of PSO, these differences arise from the kinematic and dynamic constraints of the robots as well as the independent (and therefore decentralized and asynchronous) operation of the robots. Moreover, in the implementations we do not employ reproduction steps, elimination dispersal events or cell-to-cell attraction. Still, the motion of the robots is based on a bacteria foraging behaviors. In other words,

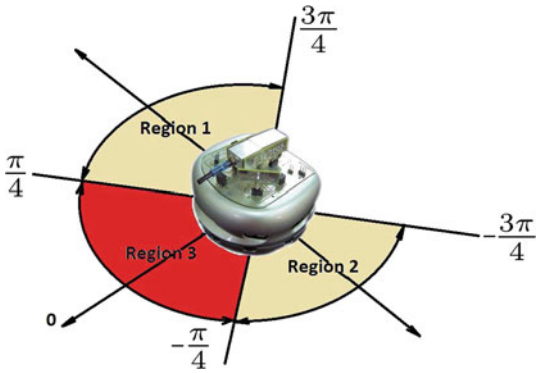


Fig. 6 Illustration of tumble and swim

the robots try to imitate bacterial chemotactic behaviors and calculate their way-points based on the BFO algorithm. At each chemotactic step while moving from their current way-points to their next way-points the robots continuously sense the chemical gas concentration and calculate the average along the path. At their next way-points they compare their readings with the average of the concentration in the traversed path. Based on the sensed information it can exhibit three different behaviors which are discussed below.

The algorithm can be briefly described by the pseudocode given in Algorithm 3, where $S(curr_i)$ refers to sensor reading in the current position of the ith robot and $S(prev_i)$ is the average of sensor reading between the starting and the ending points of the current chemotactic step. The motion between two chemotactic steps (i.e., between two way-points) is achieved using the algorithm with the simple pseudocode shown in Algorithm 2. Throughout the chemotactic steps, each robot sends continuously its own sensor readings to the base station. The experiment is continued until 50 chemotactic steps are reached.

First, the algorithm takes the absolute difference between the current concentration value $S(curr_i)$ and the calculated average value $S(prev_i)$. If the obtained value is less than the predefined constant *TresholdValue=20*, then the robots will change orientation in a random direction within the regions 1, 2, and 3 in Fig. 6 between $-\frac{3\pi}{4}$ and $\frac{3\pi}{4}$ radians. This behavior occurs when the environment is homogeneous and there is no significant change in the concentration between chemotactic steps (way-points). In contrast, if the absolute difference between $S(curr_i)$ and $S(prev_i)$ is greater than the *TresholdValue* the algorithm compares the values of $S(curr_i)$ and $S(prev_i)$ and if the sensor reading in the current step $S(curr_i)$ is greater than the average obtained during the current step $S(prev_i)$, then the robot will choose its orientation in a random direction only in region 3 shown in Fig. 6 between $-\frac{\pi}{4}$ and $\frac{\pi}{4}$ and increases randomly its swimming length between 0.2 and 0.3 m in the selected direction similar to the behavior of a bacterium in a nutrient substance swimming towards increasing concentration. In contrast, if the value of $S(curr_i)$ is less than the value of $S(prev_i)$, similar to a bacteria which has found that the new measurement is worse than the previous measurement, the robot will choose an orientation in random direction in either region 1 ($\frac{\pi}{4}$, $\frac{3\pi}{4}$) or region 2 ($-\frac{\pi}{4}$, $-\frac{3\pi}{4}$) then reduce randomly its swimming length between *0.15* and *0.2* in the selected direction.

Algorithm 3 Pseudocode of the BFO Algorithm

```
Initialize the variables
Assign the first way-points intentionally far from each other
Swim (first way-point)
Assign the second way-point randomly
Swim (second way-point)
while (Chemotactic steps N_c is not reached) do
  if (abs(S(curr_i)-S(prev_i)) < TresholdValue) then
    Tumble in all Regions(1, 2, 3) rand(-3π/4, 3π/4)
    Set swim length to 0.1 meters
  else
    if (S(curr_i) > S(prev_i)) then
      Tumble in Region 3 rand(-π/4, π/4)
      Set swim length to rand(0.2,0.3) meters
    else
      Tumble in Region 1 or in Region 2 (randomly
      chosen)
      Set swim length to rand(0.15,0.2) meters
    end if
  end if
  Compute the next way-point
  Swim (next way-point)
end while
```

Note that here as well in principle the robots execute the three tasks presented in Table 1 with slight modifications in the overall procedure. In particular, they calculate the next way-points using different strategy to explore the environment (Task 1). Moreover, the robots do not exchange information among each other or basically they do not cooperate (Task 3).

Since there is no communication and cooperation between agents they will not impose constraints on scaling the algorithm to a larger space or larger number of robots. However, in a larger area (and low robot density) there might be degradation of performance and not exploring the whole space. Therefore, the initial dispersion of the robots is important. Moreover, to overcome such shortcomings dispersal event (which are present in the original BFO algorithm proposed by Passino although not implemented here) can be implemented as well.

4.3 Ant colony optimization

In this section, the ACO algorithm and its implementation to the multi-robot search here will be described. Although the ACO algorithm was proposed to solve traveling sales person and optimal path planning problems, it has been found useful and applicable in other robotic fields as well. Dorigo's proposed algorithm has been implemented on industrial problems, telecommunication networks, dynamic and stochastic optimization problems and others [4]. The implementation here follows the ideas in [21,22]. In particular, similar to the works in [21,22], the ACO algorithm is modified and the robots are categorized in two groups which are namely ***searcher*** and ***resident*** robots. Initially, all robots have searcher identity that can communicate with other searcher robots and search for odor sources in the environment. When a robot decides that it has arrived at a position close to a maxima (i.e., close to an odor source) it becomes a resident robot. The resident robots cut off communication with other robots and move with small steps around the determined possible odor source. The ACO

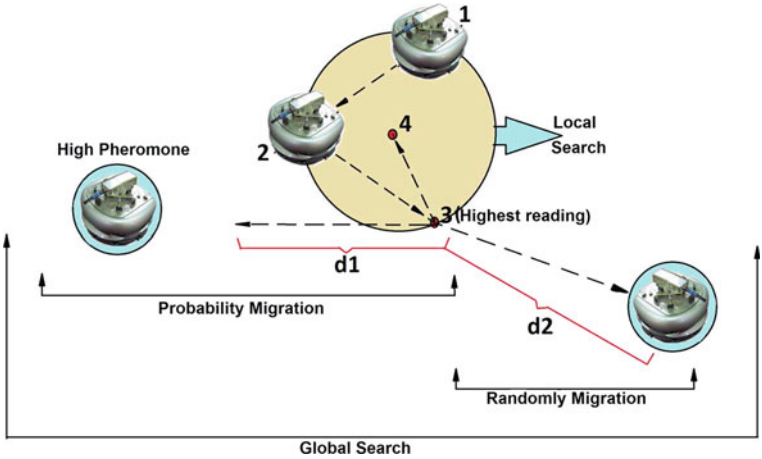


Fig. 7 Illustration of the ACO algorithm principles

search illustration is shown in Fig. 7. As shown in the illustration the modified ACO algorithm is performed in two main steps: local search and global search. At first robots perform local search, during which as can be seen in Fig. 7 all searcher robots move to the vertexes and center of gravity points of equilateral triangles and sense chemical gas concentration at each of these points. Then searcher robots compare these sensor readings and determine their next way-points using the highest concentration reading among visited points. Local search is finished when the robot reaches the determined maximum concentration point in local search. In global search, the robots communicate with each other in order to share highest sensor readings obtained during their local search steps. The searcher robots compare their own maximum concentration sensor readings to the average of the gathered searcher robots' highest readings. Based on the condition

$$C_i < C_{avg} \text{ and } q > q_0 \tag{10}$$

the searcher robots perform either random moving or probability migration. In (10) C_i refers to the current sensor reading (which is the maximum sensor reading obtained during local search), C_{avg} is the average value of the gathered sensor readings of the robots, $q_0 \in (0, 1)$ is a constant number, and q is a random number which is generated in the same range of q_0. If the condition in (10) is satisfied, the searcher robot moves randomly to find better chemical concentration regions. Otherwise, the searcher robot enters probability migration step. The probability rate equation

$$P_{ij} = \frac{\tau_j(t)^\alpha \eta_{ij}(t)^\beta}{\sum_{j=1}^{m} \tau_j(t)^\alpha \eta_{ij}(t)^\beta} \quad (i = 1, 2, \ldots, m) \tag{11}$$

is used for probability migration. In this part a given searcher robot i calculates probability rate P_{ij} for all other searchers j in the experiment/swarm and makes a decision to move towards the robot which has maximum probability rate. In Eq. (11) m is the number of searcher robots from which robot i can obtain information, $\tau_j(t)$ is the pheromone of the jth searcher robot, $\eta_{ij}(t)$ is the heuristic function given by

$$\eta_{ij}(t) = \begin{cases} e^{C_{ij}(t)/\gamma}, & j \neq i,\ j \in Searchers, \\ e^{C_i(t)/\gamma}, & j = i,\ j \in Searchers, \\ 0, & j \notin Searchers, \end{cases} \tag{12}$$

and α and β are weightiness of the pheromone and the heuristic function, respectively. In this equation $C_i(t)$ represents the current chemical concentration value of the ith searcher, $C_{ij}(t) = -(C_i(t) - C_j(t))$ is the negative difference between the chemical concentration values obtained by the ith and the jth searchers and $\gamma > 0$ is a constant. The heuristic function in (12) is one of the three heuristic functions used in [22]. It is based only on the measured chemical concentration values and the inter-agent distances do not affect the calculated values. After probability calculation searcher i takes a step towards searcher j with the highest probability rate P_{ij} based on

$$S = RD_{ij}(t) \tag{13}$$

where R is a random number in the range of $0 < R < 1$ and $D_{ij}(t)$ represents the Euclidian distance between the two searcher robots. After execution of global search, the i'th searcher robot updates it's own pheromone by means of

$$\tau_i(t+1) = (1-\rho)\tau_i(t) + \sum_{j=1}^{k} \Delta C_j(t) \tag{14}$$

where $\tau_i(t)$ is its pheromone value, $\Delta C_j(t)$ is the concentration increment value of searcher j which is approaching searcher i, and ρ is a constant which has a value between $0 < \rho < 1$

Algorithm 4 Pseudocode of the ACO Algorithm

```
Initialize the variables
Assign the first way points intentionally far from each other
Move (first way-point)
Assign the second way point randomly
Move (second way-point)
while (Max iter not reached or robot is not Resident) do
  // Local search
  Go to the nodes of the assigned pattern (corners and
  center of mass of the triangle) and read sensor data
  Go to the location of the highest sensor reading
  Send data to base station
  // Global search
  if (C_i(i) < C_avg) and (q > q_0) then
    Search in random direction
    Send data to base station
    Broadcast information to other robots
  else
    Probability Migration based on (11)-(12)-(13)
    Send data to base station
    Broadcast information to other robots
  end if
  Update Pheromone Value based on (14)
  if Resident condition is satisfied then
    Change identity to Resident Robot
  end if
end while
```

and indicates the evaporation rate of the pheromone. The pseudocode describing the operation of the algorithm is shown in Algorithm 4. Similar to the other implemented algorithms and as can be seen in Algorithm 4, at each step the robots send their own sensor readings and position information to the remote computer. Note here again that the robots still perform the three main tasks in Table 1 interfused in a different manner.

Note that in the modified ACO algorithm here communication needs among agents rise in order to calculate (10)–(14). Requiring all searchers to share information among each other may lead to shortcomings similar to those of CPSO. Therefore, similar to DAPSO a better strategy is to share information using broadcasting strategy with the searcher agents which are within communication range. This will improve the scalability of the algorithm to a larger number of robots. In the mean time, for a large space (relative to the agent communication range) it might again result in disconnected sub-groups searching in different regions of the space.

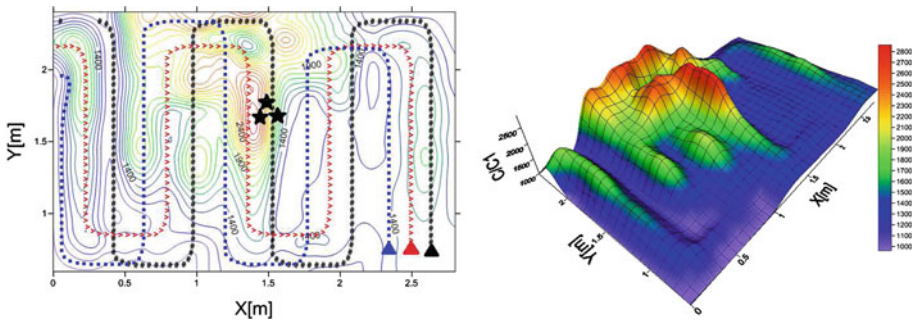
5 Experimental results

In this section we present the results obtained in the implementations considered. We performed experiments with three Khepera III and independentily with five miniQ robots in the corresponding setups discussed in Sect. 2. Initially the robots are located near a corner (as if it is an entrance) of the experimental area. The robots start the search with the 2.5 cm/s speed, which is also set as the maximum speed v_{max} in (5). Their first way-points are assigned intentionally far away from each other whereas the second way points are assigned randomly. The objective of this step is to spread the robots and cover the area as much as possible at start for better mapping of the gas concentration and better performance of the implemented algorithms. The robots communicate with the base station (a remote computer) and among each other over a wireless ad-hoc network to share their acquired information of gas concentration and position. After the two initial steps the robots start bio-inspired search as higher level path planning and determine their next way-points using the corresponding algorithm by exploiting their measurements/knowledge. The robots move from their current way-points to the next way-points by means of potential functions as described in Sect. 3. The implemented algorithms are also expected to achieve convergence of the robots into the regions with highest chemical concentration. Moreover, on the remote PC (i.e, the base station) data are gathered from the robots and a real-time 3D map of the chemical concentration is constructed. For that purpose the data received by the base station are combined, interpolated, and filtered to form the map of the gas concentration. Initially we used cubic interpolation together with a two dimensional median filter with several different masks (e.g., [3,4] and [5,5]) in Matlab.[2] Then we switched to a trial version of the *Golden Surfer 9* software as the processing and visualization tool in the base station. In Surfer we used *Kriging* as the interpolation/gridding method followed by an averaging filter with edge replicate and blank leave properties.[3]

The obtained maps are visualized in real-time as a three dimensional plot to be viewed by the operator. We performed experiments with several gas sources in different configurations. Here we present the results obtained for the case in which the sources are placed in locations close to each other in the environment. In particular, the first set of results (obtained with three Kheperas in Setup I) are for the case in which there are three sources which are placed

[2] Results obtained in Matlab are not shown.

[3] More information about surfer can be found in the Surfer manual.



(a) Trajectories of robots.

(b) 3D representation of chemical concentration.

Fig. 8 Results for the Sweep Search algorithm

Table 2 Concentration peaks of the obtained maps

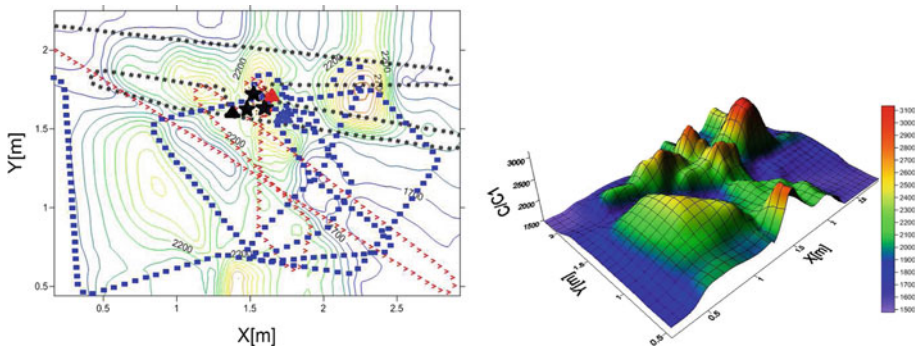
	Positions of the concentration peaks (in meters)
SWP	(1.00, 1.32), (0.98, 0.75), (1.32, 1.07), (1.59, 1.55)
DAPSO	(1.16, 1.82), (0.89, 1.25), (1.41, 0.50), (2.23, 1.80), (1.68, 1.50)
CPSO	(0.93, 0.89), (1.64, 1.09), (1.52, 1.50), (1.36, 1.86), (1.43, 1.31)
DAPSO2	(0.91, 1.45), (1.43, 1.70), (1.93, 1.68), (2.02, 1.43), (1.41, 0.75)
CPSO2	(0.22, 1.50), (0.25, 1.00), (1.57, 2.16)
BFO	(1.09, 1.07), (1.34, 1.25), (1.57, 1.16)
ACO	(1.00, 1.25), (1.07, 1.66), (1.61, 0.70)

in the environment at the locations (1.45, 1.6), (1.5, 1.7), and (1.55, 1.6). The results for the other cases are qualitatively similar (please see [23–25] for results obtained using alternative configurations).

First we present the result obtained for the case of sweeping which is shown in Fig. 8. For this case the robots traversed the predetermined paths shown in Fig. 8a. The obtained chemical concentration map is shown in Fig. 8b. Figure 8a shows also the contour map of the obtained chemical concentration map in which the bold stars represent the source locations in the environment. The colored triangle figures in the contour map are used to indicate the final positions of the robots.[4] As can be seen in Fig. 8b the highest concentration of the ethanol gas is obtained in the vicinity of the gas sources which is an intuitively expected result. This result constitutes a base or benchmark case for comparing with the results obtained using the bio-inspired search algorithms. The locations of the concentration peaks for the obtained maps including the map for sweeping in Fig. 8 are shown in Table 2.

The first implemented bio-inspired algorithm is the DAPSO algorithm the results for which are presented in Fig. 9. Figure 9a shows the trajectories of the robots superimposed on the contour plot of the extracted environmental map for an example run and the bold stars once again represent the source locations in the environment. The colored triangle figures in the contour map are again used to indicate the final positions of the robots. The paths of different robots are represented with different type of curves. The 3D plot of the obtained ethanol

[4] The same coding—bold stars = source locations, colored triangles = final robot positions—are used in all the plots in this section.



(a) Trajectories of robots.

(b) 3D representation of chemical concentration.

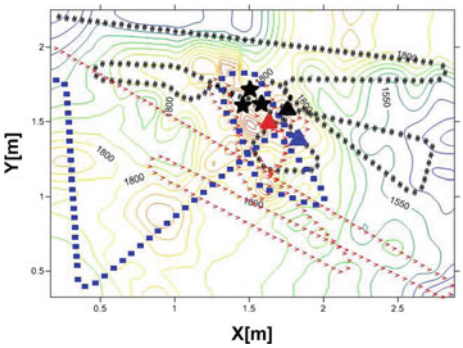
Fig. 9 Results for the DAPSO algorithm

Table 3 Final positions of robots

	Robot positions (in meters)
DAPSO	(1.36, 1.70), (1.66, 1.82), (1.70, 1.66)
CPSO	(1.66, 1.52), (1.77, 1.59), (1.86, 1.41)
DAPSO2	(1.55, 1.57), (1.75, 1.73), (1.50, 1.09)
CPSO2	(0.64, 0.50), (1.61, 0.819, (2.09, 0.70)
BFO	(1.14, 1.02), (2.07, 1.34), (1.50, 0.93)
ACO	(0.84, 1.61), (1.34, 1.73), (1.05, 1.16)

concentration in the environment is shown in Fig. 9b which has peaks at positions given in Table 2. One can see that the obtained map has its peaks in the vicinity of the chemical sources, which is similar to the sweeping case above and intuitively expected. Moreover, the robots aggregated in the area with high chemical concentration in a close distance to the contaminating sources. This was systematically the case for the DAPSO algorithm. The final positions of the robots, which can be seen as colored triangles in Fig. 9a, are also shown in Table 3. In the case in which the sources are distributed within the environment not close to each other the robots usually aggregate in the vicinity of one of the sources (results not shown). In other words, while the chemical concentration map is extracted realistically, the implemented DAPSO algorithm usually can locate only one of the contaminating sources in the environment.

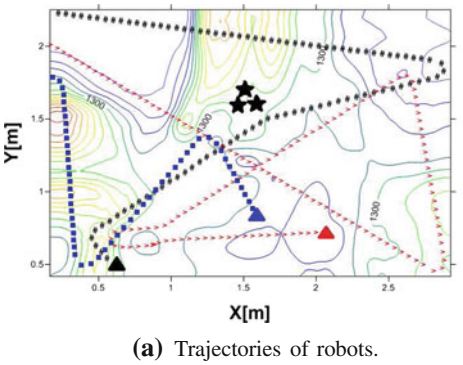
As a second algorithm we implemented CPSO the results of which are shown in Fig. 10 and the corresponding peaks and robot final positions in Tables 2 and 3, respectively. The final positions of the robots are also indicated with colored triangles in Fig. 10a. Although the results for the DAPSO and CPSO are in principle similar and in both cases the robots aggregate very close to (one of) the odor sources, the duration of the CPSO experiments usually take longer than the duration of the DAPSO experiments. The CPSO algorithm is based on synchronous interchange of information among particles. This characteristic can be considered as a disadvantage in search by multi-robot systems since the agents which complete their iteration have to wait until all agents are done. To further explore this issue we performed experiments in which we explicitly introduced agent failure. For that purpose during the search process we intentionally stop one robot and cut off its communication with



(a) Trajectories of robots.

(b) 3D representation of chemical concentration.

Fig. 10 Results for the CPSO algorithm



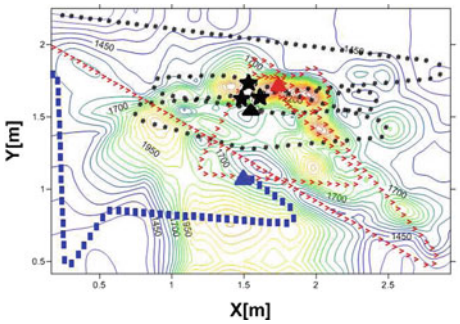
(a) Trajectories of robots.

(b) 3D representation of chemical concentration.

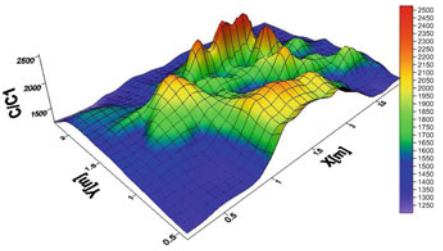
Fig. 11 Results for the CPSO algorithm—one agent fails during run

the other robots. Figure 11 shows the result for that case obtained using the CPSO algorithm. For this case once the other (unfailed) robots reach their own current way points they are stuck and the search stalled since they wait for information from the failed robot. Therefore, they fail to aggregate in the vicinity of (and therefore to locate) a contaminating source. These can be seen from the trajectories of the robots in Fig. 11a, where the colored triangles indicate the final positions of the robots. Moreover, the obtained chemical concentration map shown in Fig. 11b is less accurate compared to the previous cases. The peaks of the map and the robot positions are given in Tables 2 and 3, respectively, where the results for this case are labeled as CPSO2. The robot positions are the positions at which the system was stalled.

To compare with CPSO, we introduced the same failure during a run of the DAPSO algorithm as well the results for which are shown in Fig. 12 with the corresponding concentration peaks and robot positions in Tables 2 and 3 with the label DAPSO2. In contrast to the CPSO algorithm, in the DAPSO algorithm the remaining (unfailed) robots were able to continue their missions and construct a more accurate map despite the failure of one of the agents. Moreover, the two unfailed robots completed their iterations in the vicinity of the contaminating source and were able to locate it. The main reason for the success of the DAPSO

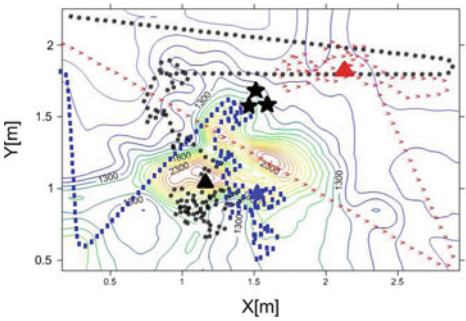


(a) Trajectories of robots.

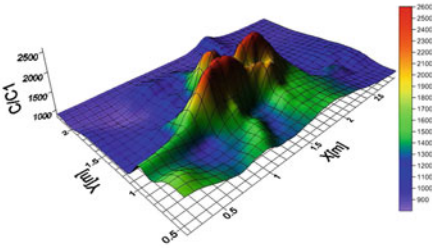


(b) 3D representation of chemical concentration.

Fig. 12 Results for the DAPSO algorithm—one agent fails during run



(a) Trajectories of robots.

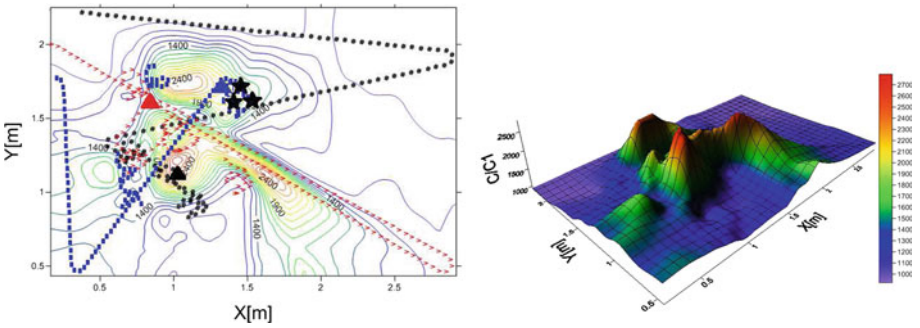


(b) 3D representation of chemical concentration.

Fig. 13 Results for the BFO algorithm

algorithm is its decentralized and asynchronous operation and the fact that the robots do not have to wait for the other robots to advance into their subsequent iterations.

In BFO, which is the next implemented algorithm, the agents do not exchange information and therefore there is no cooperation between the agents. Is it possible to implement cell-to-cell attraction to facilitate swarming and aggregation. However, in this article we did not implement such property. Therefore, in the implementation here during search the robots send their position information and sensor measurement only to the base station. The results for this case are shown in Fig. 13. As in the previous cases the paths traversed by the robots are shown in Fig. 13a superimposed on the contour plot of the obtained chemical concentration map, whereas the $3D$ plot of the obtained map is shown in Fig. 13b. The peaks of the obtained map and the positions of the robot are also given in Tables 2 and 3, respectively. Since the robots operate based on a bacteria inspired greedy strategy and there is no communication and cooperation between them they do not aggregate in a common location. However, this results in the ability to locate more than one maxima of the profile. In fact, as can be seen from Fig. 13a at the end of the experiment two of the robots were located at positions which they perceive as local maximum concentration points. These points seem little bit displaced from the sources due to possible airflow from the open corner of the arena. In the experiments with different configuration of the sources located far from each other (results not shown)



(a) Trajectories of robots.

(b) 3D representation of chemical concentration.

Fig. 14 Results for the ACO algorithm

Table 4 Final distance of robots to nearest concentration peaks

	DAPSO	CPSO	DAPSO2	CPSO2	BFO	ACO
R1	0.23	0.14	0.18	0.63	0.06	0.23
R2	0.32	0.27	0.19	1.34	0.53	0.28
R3	0.16	0.35	0.35	1.55	0.24	0.10
Av.	0.24	0.25	0.24	1.17	0.28	0.20

the robots were mostly able to locate more than one source, which is in contrast to the DAPSO (and CPSO) algorithm where the robots usually aggregate around a single source. In both algorithms the robots were able to build a reasonably realistic 3D map of the chemical concentration.

As one would recall from Sect. 4.3 in the ACO algorithm the robots exchange pheromone information and cooperate as long as they are searcher robots. Once a robot becomes a resident robot, it cuts off communication and stops the search as well as the cooperation with the other robots. This type of behavior is somewhere in between the continuous cooperation in the PSO and no cooperation in the BFO algorithms. The results obtained using the ACO algorithm are shown in Fig. 14 with corresponding concentration peaks and robot positions in Tables 2 and 3, respectively. As can be seen from Fig. 14a the robots finish at different locations close to what they perceive as the "high" concentration points. Therefore, similar to the case of BFO, in the ACO algorithm the robots are able to locate more than one source.

Table 4 shows the distance of the robots[5] to the nearest concentration peaks in the obtained map. As can be seen from the table all the algorithms except CPSO under failure show similar performance in terms of locating robots close to at least one concentration peak.

Table 5 shows the distance of the robots to the nearest chemical sources. Once again it is seen that CPSO under robot failure performs worst. Among the remaining cases it is seen that PSO based algorithms perform better than the other cases including the DAPSO algorithm under robot failure.

To evaluate the map building performance of the algorithms we analyzed the distance of the nearest concentration peaks of the obtained maps to the chemical sources whose results

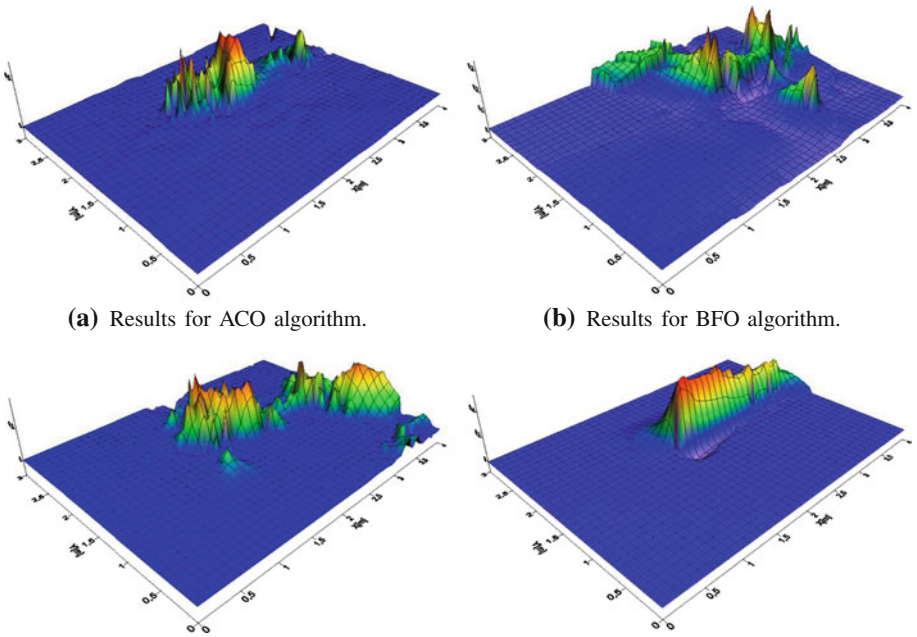
[5] The distances in Tables 4, 5, and 6 are in meters.

Table 5 Final distance of robots to nearest chemical sources

	DAPSO	CPSO	DAPSO2	CPSO2	BFO	ACO
R1	0.14	0.13	0.03	1.37	0.66	0.60
R2	0.20	0.22	0.24	0.78	0.58	0.16
R3	0.17	0.36	0.51	1.05	0.67	0.60
Av.	0.17	0.24	0.26	1.07	0.64	0.46

Table 6 Distance of concentration peaks to nearest chemical sources

	DAPSO	CPSO	DAPSO2	CPSO2	BFO	ACO	SWP
S1	0.25	0.12	0.10	0.57	0.37	0.39	0.15
S2	0.27	0.20	0.07	0.46	0.48	0.43	0.18
S3	0.17	0.10	0.16	0.56	0.41	0.49	0.07
Av.	0.23	0.14	0.11	0.53	0.42	0.44	0.13



(a) Results for ACO algorithm.

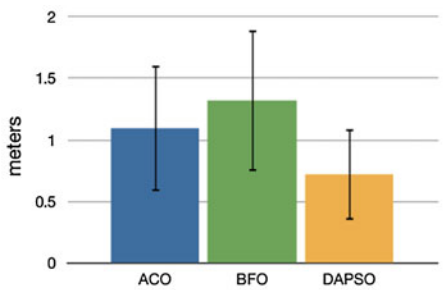
(b) Results for BFO algorithm.

(c) Results for DAPSO algorithm.

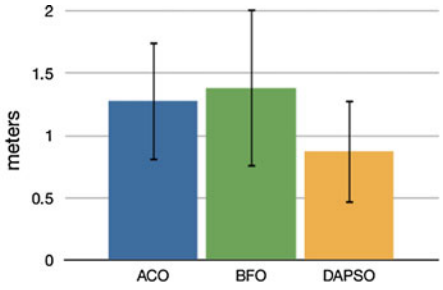
(d) Ideal plume map (obtained through simulation).

Fig. 15 Maps obtained with miniQs in Setup II

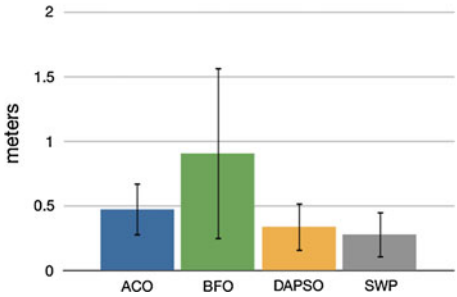
are shown in Table 6. The result for the sweeping is also shown in the table. Although the algorithms (excluding CPSO under failure) have comparable performance one can see that that PSO based algorithms showed better performance than the BFO and the ACO algorithms in the above implementations.



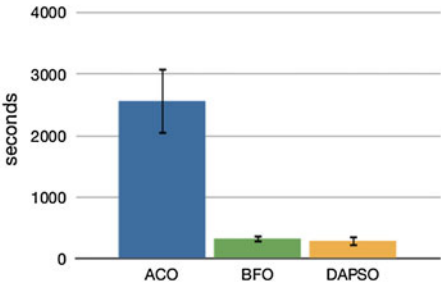
(a) Average final distance of robots to nearest concentration peaks.



(b) Average final distance of robots to the chemical source.



(c) Distance of concentration peaks to the chemical source.



(d) Task completion times.

Fig. 16 Results with miniQs (averages over 5 runs)

In order to further investigate the performance of the BFO, ACO, and DAPSO algorithms we performed more experiments in Setup II using five miniQ robots. Example maps obtained in these experiments are shown in Fig. 15. Note that in these experiments there is single source positioned at (1.5, 1.8) and an airflow created by fans mounded on one of the side walls of the arena. Under these conditions the ideal map obtained through simulation is shown in Fig. 15d. Given the imperfections in real experiments one can easily state that the obtained maps are realistic. They are also in principle similar to the maps presented above except for differences due to the airflow. The cumulative results for five experiments are summarized in Fig. 16. Figure 16a and b show the average final distance of the robots to the nearest concentration peaks and to the chemical source, respectively. The vertical lines over the bars show the minimum and the maximum distances. These plots provide information about the source localization performances of the algorithms. One can easily see from the figures that the DAPSO algorithm has outperformed both the ACO and the BFO algorithms. Figure 16c shows the average distance of the concentration peaks to the chemical source and provides information about the mapping performances of the algorithms. Results for a sweep search are also included in the figure. As can be seen from the figure once again the DAPSO algorithm performs better than both the ACO and the BFO algorithms. These results are also consistent with the maps in Fig. 15. Figure 16d shows the recorded task completion times for the algorithms. It can be seen that BFO and DAPSO converge fast, whereas the ACO algorithm runs for a much longer time. Taking into account all the plots one can state that among the considered algorithms the DAPSO algorithm showed the best performance

in the performed experiments. However, to generalize such a conclusion will need more comprehensive investigation and further research.

It is worth mentioning that we are not compensating the gas sensors' response time. This will act as a low pass filter with rise time of about 2 s and recovery time of 10–15 s. Considering that all experiments were done with the robots controlled by the sensed potential field, moving at a maximum velocity of 2.5 cm/s, in a worst case scenario, in environments containing step concentration changes (an unreal case due to the physics of odor transport), the time response of the sensors would be responsible for an error of about 5 cm in the detection of a step and about 35 cm in the detection of its finishing. In a real testing environment, similar to the ones of the experiments, with smooth concentration changes and the robots crossing the area in multiple directions, we can say that the sensors' response time should be responsible for errors of 5–20 cm.

Although the considered algorithms are designed (and more suitable) for search for minima or maxima (e.g. concentration peaks) it is seen from the results that they perform sufficiently well for the case of mapping as well. This shows they lead to sufficient exploration of the environment during search although there is no intelligent/intentional bias to visit privilege areas of weaker knowledge. Moreover, they are easy to implement and are computationally simple. This constitutes an advantage over more complex algorithms.

6 Concluding remarks

In this study chemical concentration map building and odor source localization using bio-inspired search algorithms such as DAPSO, Canonical Particle Swarm Optimization (CPSO), BFO, and ACO by multiple mobile robots was considered. The results show that the robots succeed in extracting a three dimensional map of the chemical gas concentration. Moreover, the robots are able to locate in the vicinity of (and therefore to determine) the contaminating leaks (which are the regions which they perceive as areas of high concentration). While in the PSO based algorithms the robots mostly aggregate in one location in the BFO and the ACO algorithms they can finish the search at different locations (and therefore be able to locate more than one peak or source). The implementations are realized in a laboratory setting using miniature mobile robots equipped with alcohol sensors and real ethanol gas as chemical contaminant. In our implementations the DAPSO algorithm showed slightly better overall performance compared to the other considered algorithms. However, in order to generalize such a conclusion there is a need for further thorough investigation. It is known and has already been studied that the performance of odor source localization algorithms, in environments with relevant airflow, can be improved taking into account the airflow's direction and tracking a detected odor plume in the upwind direction until its source. Although there was some airflow in part of the experiments, concentrating more on the map building task we did not consider wind and turbulence effects in our implementations. Using anemometers in the robots and adapting the algorithms in order to take into account the airflow can be a topic of future research. Moreover, establishing a resident sensing grid system capable of extracting the instantaneous map of the contaminating chemical into the experimental arena for better comparison of the performance of the algorithms is a possible future research directions. Equipping the robots with a sensor array and determining the composition of the chemicals in addition to the other missions is another possibility for future extension. Comparison of the performance of the bio-inspired algorithms with the performance of conventional methods as well as considering more complex environments are also potential directions of research.

Acknowledgments Part of this work was performed at the time V. Gazi was affiliated with and M. Kırtay was a visiting intern at TOBB ETU. The authors would like to thank Yunus Ataş, Pedro Sousa, and Bruno Antunes for their help in software and hardware development during various stages of this study.

References

1. Oyekan, J., Hu, H., & Gu, D. (2009, December). Exploiting bacteria swarms for pollution mapping. In *IEEE International Conference on Robotics and Biomimetics* (pp. 39–44). Guilin, China.
2. Sierakowsk, C. A., & Coelho, L. S. (2006). Path planning optimization for mobile robots based on bacteria colony approach. *Applied soft computing technologies: The challenge of complexity*, *34*, 187–198.
3. Wu, C., Zhang, N., Jiang, J., Yang, J., & Liang, Y. (2007, April). Improved bacterial foraging algorithms and their applications to job shop sheduling problems. In *International Conference on Adaptive and Natural Computing Algorithms. Series: Lecture notes in computer science, Vol. 4431* (pp. 562–569). Warsaw, Poland.
4. Dorigo, M., Birattari, M., & Stützle, T. (2006). Ant colony optimization artificial ants as a computational intelligence technique. *IEEE Computational Intelligence Magazine*, *1*(4), 28–39.
5. Hayes, A. T., Martinoli, A., & Goodman, R. M. (2002). Distributed odor source localization. *IEEE Sensor Journal*, *2*(3), 260–271.
6. Farrell, J. A., Pang, S., & Li, W. (2003). Plume mapping via hidden markov methods. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *33*(6), 850–863.
7. Pang, S., & Farrell, J. A. (2006). Chemical plume source localization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, *36*(5), 1068–1080.
8. Lilienthal, A., & Ducket, T. (2004). Building gas concentration gridmaps with mobile robot. *Robotics and Autonomous Systems*, *48*(1), 3–16.
9. Loutfi, A., Coradeschi, S., Lilienthal, A. J., & Gonzalez, J. (2009). Gas distribution mapping of multiple odour sources using a mobile robot. *Robotica*, *27*(2), 311–319.
10. Kennedy, J., & Eberhart, R. C. (1995). Particle swarm optimization. In *IEEE International Conference on Neural Networks* (pp. 1942–1948).
11. Marques, L., Nunes, U., & de Almedia, A.T. (2004, September) Finding odours across large search spaces: A particle swarm-based approach. In *International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR2004)* (pp. 419–426). Madrid, Spain: Springer.
12. Marques, L., Nunes, U., & de Almedia, A. T. (2006). Particle swarm-based olfactory guided search. *Autonomous Robots*, *20*(3), 277–287.
13. Pugh, J., & Martinoli, A. (2007). Inspiring and modeling multi-robot seach with particle swarm optimization. In *IEEE Swarm Intelligence Symposium (SIS 2007)*. Honolulu, USA.
14. Pugh, J., & Martinoli, A. (2008). Distributed adaptation in multi-robot search using particle swarm optimization. In *10th International Conference on the Simulation of Adaptive Behavior* (pp. 393–402). Osaka, Japan: Springer.
15. Hereford, J. M. (2006). A distributed particle swarm optimization algorithm for swarm robotics application. In *Congress on Evolutionary Computation* (pp. 6143–6149). Vancouver, Canada.
16. Hereford, J. M., & Siebold, M. (2008). Multi-robot search using a physically-embedded particle swarm optimization. *International Journal of Computational Intelligence Research*, *4*(2), 179–209.
17. Doctor, S., Venayagamoorthy, G. K., & Gudise, V. G. (2004). Optimal PSO for collective robotics search applications. In *IEEE Congress on Evolutionary Computation* (pp. 1390–1395). Portland, USA.
18. Passino, K. M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine*, *22*(3), 52–67.
19. Berg, H. (2004). *E. coli in motion*. New York: Springer.
20. Dhariwal, A., Sukhatme, G. S., & Requicha, A. A. (2004, April). Bacterium-inspired robots for environmental monitoring. In *IEEE International Conference on Robotics and Automation* (pp. 1436–1443). Louisiana, USA.
21. Meng, Q.-H., Li, J.-C., Li, F., & Zeng, M. (2006, December). Mobile robots odor localization with an improved ant colony algorithm. In *IEEE International Conference on Robotics and Biomimetics* (pp. 959–964). Kunming, China.
22. Zou, Y., Luo, D., & Chen, W. (2009, December) Swarm robotic odor source localization using ant colony algorithm. In *IEEE International Conference on Control and Automation* (pp. 792–796). Christchurch, New Zealand.
23. Turduev, M., Atas, Y., Sousa, P., Gazi, V., & Marques, L. (2010, October). Cooperative chemical concentration map building using decentralized asynchronous particle swarm optimization based search

algorithm by mobile robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)* (pp. 4175–4180). Taipei, Taiwan.
24. Turduev, M., Kırtay, M., Sousa, P., Gazi, V., & Marques, L. (2010, October) Chemical concentration map building through bacterial foraging optimization based search algorithm by mobile robots. In *IEEE International Conference on Systems, Man, and Cybernetics (SMC 2010)* (pp. 3242–3249). Istanbul, Turkey.
25. Kırtay, M., Turduev, M., Sousa, P., Gazi, V., & Marques, L. (2010, September). Chemical concentration map building through ant colony optimization based search algorithm. In *Turkish National Conference on Automotic Control (TOK 2010)* (pp. 180–186). Kocaeli, Turkey (in Turkish).
26. Pascoal, J., Sousa, P., & Marques, L. (2008, September). Khenose—a smart transducer for gas sensing. In *International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines (CLAWAR2008)* (pp. 993–1000). Coimbra, Portugal: World Scientific Press.
27. Borenstein, J., & Feng, L. (1995, August). Correction of systematic dead-reckoning errors in mobile robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 95)* (pp. 569–574). Pittsburgh, USA.
28. Borenstein, J., & Feng, L. (1995, October). Umbmark: A benchmark test for measuring deadreckoning errors in mobile robots. In *SPIE Conference on Mobile Robots*. Pennsylvania, USA.
29. Atas, Y. (2010). *Localization and mapping in mobile robot systems*. Unpublished master's thesis, TOBB University of Economics and Technology. Turkey (in Turkish).
30. Marques, L., Almeida, N., & de Almeida, A. T. (2003, October). Olfactory sensory system for odour-plume tracking and localization. In *IEEE International Conferance on Sensors* (pp. 418–423). Toronto, Canada.
31. Almeida, N., Marques, L., & de Almeida, A. T. (2003). Fast identification of gas mixtures through the processing of transient responses of an electronic nose. In *EuroSensors*. Guimaraes, Portugal.
32. Barsan, N., & Tomescu, A. (1995). Calibration procedure for SnO_2-based gas sensors. *Thin Solid Films*, *259*(1), 91–95.
33. Samiloglu, A. T., Gazi, V., & Koku, A. B. (2008). Comparison of three orientation agreement strategies in self-propelled particle systems with turn angle restrictions in synchronous and asynchronous settings. *Asian Journal of Control*, *10*(2), 212–232.
34. Clerc, M., & Kennedy, J. (2002). The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, *6*(1), 58–73.
35. Akat, S. B., Gazi, V., & Marques, L. (2010). Asynchronous particle swarm optimization based search with a multi-robot system: Simulation and implementation on real robotic system. *Turkish Journal of Electrical Engineering and Computer Sciences*, *18*(5), 749–764.
36. Akat, S. B., & Gazi, V. (2008, September). Particle swarm optimization with dynamic neighbourhood topology:three neighborhood strategies and preliminary results. *In IEEE Swarm Intelligence Symposium (SIS-2008)*. St. Louis, USA.
37. Akat, S. B., & Gazi, V. (2008, September). Decentralized asynchronous particle swarm optimization. In *IEEE Swarm Intelligence Symposium (SIS-2008)*. St.Louis, USA.