On-board Evolutionary Algorithm and Off-line Rule Discovery for Column Formation in Swarm Robotics

Asuki Kuno, Jean-Marc Montanier, Shigeru Takano, Nicolas Bredeche, Marc Schoenauer, Michèle Sebag, Enoshin Suzuki

On-board Evolutionary Algorithm and Off-line Rule Discovery for Column Formation in Swarm Robotics

Asuki Kouno
Grad. School of Systems Life Sciences
Kyushu University
819-0395 Fukuoka, Japan
Email: 3sl10004n@sls.kyushu-u.ac.jp

Jean-Marc Montanier
TAO - Univ. Paris-Sud, INRIA, CNRS
LRI, Bat. 490, Univ. Paris-Sud,
F-91405 Orsay, France
Email: montanier@lri.fr

Shigeru Takano
Dept. Informatics, ISEE
Kyushu University
819-0395 Fukuoka, Japan
Email: takano@inf.kyushu-u.ac.jp

Nicolas Bredeche
TAO - Univ. Paris-Sud, INRIA, CNRS
LRI, Bat. 490, Univ. Paris-Sud,
F-91405 Orsay, France
Email: nicolas.bredeche@lri.fr

Marc Schoenauer
TAO - INRIA & Univ. Paris-Sud
LRI, Bat. 490, Univ. Paris-Sud,
F-91405 Orsay, France
Email: marc.schoenauer@inria.fr

Michèle Sebag
TAO - CNRS & Univ. Paris-Sud
LRI, Bat. 490, Univ. Paris-Sud,
F-91405 Orsay, France
Email: michele.sebag@lri.fr

Einoshin Suzuki
Dept. Informatics, ISEE
Kyushu University
819-0395 Fukuoka, Japan
Email: suzuki@inf.kyushu-u.ac.jp

***Abstract*—This paper aims at building autonomous controllers for swarm robots, specifically aimed at enforcing a given shape formation, here a column formation. The proposed approach features two main characteristics. Firstly, a state-of-the-art evolutionary setting is used to achieve the on-board optimization of the controller, avoiding any simulator-based approach. Secondly, as the cost of physical experiments might be prohibitively high for plain evolutionary approaches, a data mining approach is achieved on the top of evolution; rule discovery is used to discover the most promising regions in the controller search space. The merits of the approach are experimentally validated using a 5 robot formation, showing that the hybrid evolutionary learning process outperforms evolution alone in terms of swarm speed and shape quality.**

I. Introduction

This paper is interested in swarm robotics, and specifically in swarm shape formation. How to enforce coordinated moves in a decentralized and distributed manner has mostly been studied in the literature from the perspective of complex system description, in the sense that this study focuses on the macroscopic shape produced through a few rules [1].

The focus here is on complex system design: our goal is to identify simple autonomous robot controllers resulting in a given swarm shape. The complexity arises from the need for a spatial coordination of the group of robots, which is expected to result in their column formation. The challenge is to reach this goal *in situ*, i.e., by directly training the controllers on the physical robots as opposed to *in silico*, i.e., by training the controllers on a robot simulator. The motivations behind on-board training are twofold. On the one hand, simulator-based controller design is known to result in the so-called reality gap [2], that is, the *in-situ* performance of the controller might be arbitrarily worse than that *in silico*. Quite a few approaches have been proposed to sidestep the reality gap, through, e.g., closing the gap [3] or using dimensionality reduction [4]. On the other hand, the presented study resumes an earlier work, devoted to building low-cost robots [5]: there is no reliable simulator available for these robots, and it makes sense to design a training process relevant to such simulator-less experimental frameworks.

The main drawback of on-board training is twofold. On the one hand, it takes much longer than using the simulator; 12 controller architectures have been considered in [5], and training each one of them in order to select the best one takes a large amount of time. On the other hand, in-situ training is physically demanding for the robots and might entail physical hazards.

The approach proposed in this paper to keep the training time within reasonable time while obtaining a good controller works as follows. Firstly, an evolutionary on-board approach, initially developed in the SYMBRION framework [6], [7] and referred to as (1+1) Restart-Online Adaptation Algorithm (ROAA)[1], is used to explore the controller search space online

[1]Here 1+1 means the champion versus the challenger, because the algorithm is essentially an iterative refinement by competitions between them.

without communication[2]. Secondly, the resulting behavior of the robots is recorded and the video, referred to as robotic log, is exploited using supervised machine learning (ML) [8]; the ML step produces rules characterizing desirable behaviors; these rules are used to constrain the search space explored by ROAA and thus speed up the search. The experimental validation of the approach confirms that satisfactory solutions can be obtained with a significant reduction of the experimental cost.

The core question of this paper is designing self-adapted multi-robot systems using rule discovery, a data mining method. Relevant works exist in the topic of learning policies, rules, or parameters in the context of multi-robot systems. For instance, [9] proposes pure reinforcement learning techniques using small groups of robots, i.e., up to 10 robots, on classic tasks such as foraging. In [10], reinforcement learning is used in team coordination tasks such as RoboCup setups. [11] tackles classic evolutionary robotics in the context of swarms, where an optimization is executed off-line and then the result is used as the solution. On the contrary, [12] is recognized as the seminal work in embodied evolutionary robotics, as the optimization is executed on-board, i.e., while the robots are running. Compared to these works, our approach discovers explicit, declarative statements, i.e., rules, in terms of the controllers of the swarm robot, providing a clearer view to the human designers.

The rest of this paper is structured as follows. In Section II, we explain the problem of constructing robots that move in column formation. In Section III, we describe our evolutionary approach for the problem. Section IV is devoted to our integration of rule discovery into our framework. We show our experimental evaluation in Section V, and conclusions and future work in Section VI.

II. Robots Moving in Column Formation

A. Motivations for the Column Formation Task

As mentioned in the previous section, various tasks ranging from patrolling to residue disposal have been achieved by swarms in the literature either in simulation or in situ [13], [14]. The simple task of column formation, i.e., each robot in the swarm aims at following another robot, resulting in one or several lines of moving robots, was selected for the following reasons. On the one hand, it involves the main two skills for group behavior, namely the ability to detect another member of the swarm and to keep the contact while moving. On the other hand, the column formation task can be easily assessed with respect to two instant criteria: the overall shape of the formation (which can involve several leaders) and its overall speed. Another second order criterion is related to the presence of deadlocks: robots follow a circle and there are no leaders.

The ability to easily assess all the above three criteria is the cornerstone of the presented work, since our goal is to demonstrate a robot training procedure which does not rely on simulations.

[2]Currently our swarm robots do not communicate each other, prohibiting the use of evolutionary algorithms which require an exchange of genomes, e.g., genetic algorithms.

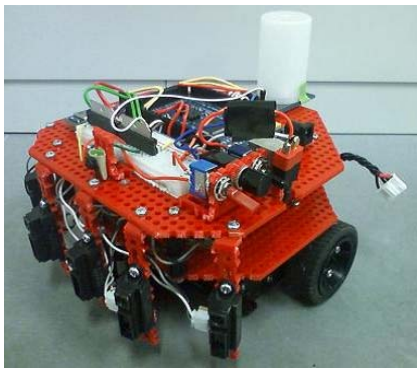

Fig. 1. The physical robot used in the experiments. Its identification in the robotic log video is possible as every robot is covered by a colored sheet of paper (see text).

B. Robot Architecture

We believe that details of our physical robots are important as we tackle an on-board optimization of the controller, Resuming an earlier work [5], each robot (Fig. 1) is equipped with 5 infrared (IR) proximity sensors, a camera, a low-cost micro-processing unit (MPU), and a LED light signaling its presence to the other robots. Due to the limited size of the RAM of the MPU and the real-time constraint, our robot processes 20×15 pixel images captured by its camera. Additionally, each robot is equipped with a Bluetooth unit, enabling it to send its internal state to the central server during the experiments for analysis facility. It must be emphasized that the robot is fully autonomous: it only uses the signal from its IR sensors and camera and does not involve a positioning module such as GPS or RFID.

Each IR sensor measures the distance to an obstacle by emitting an ultra-red light and reading its reflection from the obstacle as a signal value. Clearly, the measurement is subject to noise and each sensor might fail to detect an obstacle which is too small, and/or absorbs or diffuses the red light.

The low resolution camera is essentially used to detect the LED light of the other robots. The LED unit is a printed circuit board with a LED device stored in a camera-film case. It can emit three colors (red, blue, and green) and only the green color will be used in the experiments.

The robot locomotion uses two wheels and a supporting ball, enforcing a sufficient robustness of the robot (in contrast to our former use of two caterpillar [5]). The driving system consists of two motors each of which is connected to a wheel, a controlling circuit, a gear box, and the supporting ball. The driving system allows the robot to move forward or backward, to stop, and to make a pivot turn. For cost reasons, only digital control is used. The robot controller provides the motor commands based on its sensor input and a set of thresholds (see Table I).

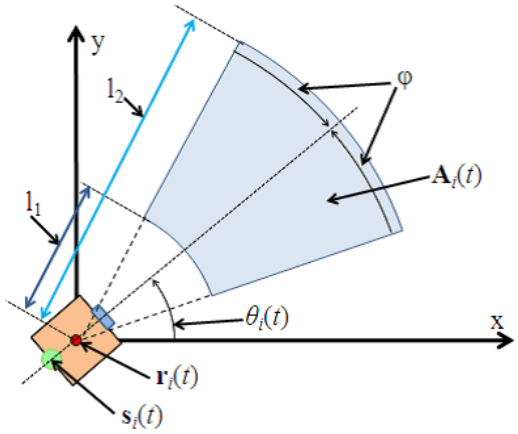

Fig. 2. Notations: position and perception range of robot i at time t

C. Hidden and Observable Data

Data concerning an experiment may be divided into hidden and observable data to the designer. The hidden data are images taken by the camera of the robot and the readings of its IR sensors. They are called hidden data because they are available to the robot but not to the designer. The observable data are generated from video filmed by a USB camera located on the ceiling and processed by an external PC. The USB camera films the 2D indoor field where the robots move. Note that the observable data and the results of its processing are available to the designer.

In the field, each robot (Fig. 1) is covered by a colored sheet of paper and is associated to a distinct color, enabling its identification in the video. From the video, it is thus possible to estimate the center of the location of robot i at time t, noted $\mathbf{r}_i(t)$, as the center of mass of the pixels of the i-th color. The center of the location of its LED unit (identified as the closest LED unit[3]) is denoted $\mathbf{s}_i(t)$, as shown in Fig. 2. Occasionally $\mathbf{s}_i(t)$ is not detected due to illumination problems and in such a case $\mathbf{s}_i(t)$ is set to $\mathbf{s}_i(t-1)$.

The x and y coordinates of $\mathbf{r}_i(t)$ and those of $\mathbf{s}_i(t)$ are denoted by $\mathbf{r}_i(t)_x$, $\mathbf{r}_i(t)_y$, $\mathbf{s}_i(t)_x$, and $\mathbf{s}_i(t)_y$, respectively. The direction $\theta_i(t)$ of robot i can thus be estimated from the angle of $\mathbf{r}_i(t) - \mathbf{s}_i(t)$ with the horizontal line, computed as:

$$\theta_i(t) = \arctan\left(\frac{\mathbf{r}_i(t)_y \ - \ \mathbf{s}_i(t)_y}{\mathbf{r}_i(t)_x \ - \ \mathbf{s}_i(t)_x}\right) \tag{1}$$

In accordance with preliminary experiments, the vision range of robot i denoted $\mathbf{A}_i(t)$ is a truncated cone in the $\theta_i(t)$ direction, depicted as the shaded area in Fig. 2. The robot neither sees too close obstacles (distance less than l_1) nor too far away ones (distance greater than l_2). The vision angular scope is φ: only obstacles in the angular sector $[\theta_i(t) - \varphi, \theta_i(t) + \varphi]$ can be detected (provided that their distance to the robot is in $[l_1, l_2]$).

[3]Recall that LED light is used to signal the presence of the robot to other robots.

D. Task definition and assessment criteria

The swarm involves n robots, all equipped with an instance of the same controller. We assume that the n robots form K columns, where K represents the number of columns.

From the observable data, some other parameters can be computed; all definitions are related to a specific time step t, which is omitted for the sake of brevity when it is clear from the context.

- Robot i follows robot k at time t if and only if the center of location of the LED unit of robot k belongs to the perception range of robot i ($\mathbf{s}_k(t) \in \mathbf{A}_i(t)$).
- Robot i is a leader if i) it does not follow any other robot; ii) there exists at least one robot following it.
- A column is composed of n' robots $j_1, j_2, \ldots j_{n'}$ such that robot j_{i+1} follows robot j_i and robot j_1 is a leader, where $n' \leq n,\ 0 \leq i \leq n'-1$.
- A deadlock is composed of n' robots $j_1, j_2, \ldots j_{n'}$ such that robot j_{i+1} follows robot j_i and robot j_1 follows robot $j_{n'}$, where $n' \leq n,\ 0 \leq i \leq n'-1$.

The quality of the controller is measured from the quality of the swarm, averaged over the time-length T of the experiment. Two criteria are computed from the above definitions.

The first criterion measures the cohesiveness of the swarm. Let $\mathcal{F}(i,t)$ denote the number of robots following (a follower of) robot i at time t, with $\mathcal{F}(i,t)$ being set to 0 if robot i is not a leader. Then the pursuit rate F is defined as the fraction of follower robots, averaged over the experiment time interval $[0,T]$:

$$F = \frac{1}{(n-1)T}\sum_{t=1}^{T}\sum_{i=1}^{n}\mathcal{F}(i,t) \tag{2}$$

F reaches its maximum ($F=1$) when all robots are engaged in a single column, i.e., $K=1$. On the other hand, as the value of K increases, F typically decreases because $\mathcal{F}(i,t)$ cannot take a large value in the presence of K leaders. Robots in a deadlock are not accounted for.

The second criterion measures the efficiency of the swarm, estimated from the speed of the leaders. Formally, let $v_i(t)$ be defined as the distance between the position of robot i at time $t+1$ and t, i.e., $v_i(t) = ||\mathbf{r}_i(t+1) - \mathbf{r}_i(t)||$, then the swarm efficiency is defined as the average leader speed, weighted by its number of followers:

$$V = \frac{1}{F}\sum_{t=1}^{T}\sum_{i=1}^{n}v_i(t)\mathcal{F}(i,t) \tag{3}$$

Both criteria thus measure the overall accuracy of the controller. If F is high the cohesiveness is good but the robots might be moving very slowly. At the other extreme, V might be high because of one leader robot moving fast and followed by a few followers while the other robots are engaged in a deadlock. If both criteria score high, however, then the controller does demonstrate its ability to detect and follow a fast moving target in an autonomous way. Note that similar proposals exist in collective robotics [15] and social

animals [16], though our criteria are specific to the column formation task and cannot be derived from those for more general settings [15], [16].

III. Evolutionary Approach for the Problem

A. Reactive Controller

The control program, which makes no assumption about its environment, enables a robot to move basically forward by avoiding the perimeters if it is a leader or to follow a preceding robot. Note that we use a simplified version of the controllers used in [5], but we evolve it using the (1+1) ROAA [7], which we explain in the next section, during the experiments.

The reactive controller at a specific time frame is given in Table I. A controller is a function of $P_{\mathrm{LED}} \times R_{\mathrm{IR}} \rightarrow M \times N$, where $P_{\mathrm{LED}}, R_{\mathrm{IR}}, M, N$ represent the domain of the position of the preceding robot, the domain of the maximum reading value of the IR proximity sensor, the domain of the kind of motion, and the domain of the time of rotation, respectively. The preceding robot is indicated by the x-coordinate x_{image} of its LED unit along the horizontal axis of the image. We use a parameter α ($1 \leq \alpha \leq 10$) to divide the axis into 3 bins, the 2 segmenting points being α and $19-\alpha$. P_{LED} consists of the 3 bins and a null value. Basically the robot tries to locate the preceding robot in the middle bin by appropriate turns.

The closer an obstacle is, the larger is the reading value of the IR proximity sensor. Thus the maximum reading value of the IR proximity sensor may be used to estimate the distance to the closest object which possibly corresponds to the preceding robot. We use another parameter β ($18 \leq \beta \leq 30$) to divide the estimated distance x_{IR} to the closest object into 3 bins. R_{IR} consists of the 3 bins. Basically the robot tries to avoid collisions if the distance is below β_0 cm (fixed to 18 cm in the experiments) and tries to locate the preceding robot in the range of β_0 cm - β cm.

M consists of {move forward, move backward, stop, turn left, turn right}. The robot stops when $\alpha < x_{\text{image}} < (19-\alpha)$ and $\beta_0 \leq x_{\mathrm{IR}} < \beta$, N is defined as a set of real values when $M = \{\text{turn left, turn right}\}$ and otherwise is undefined. The third parameter we employ is γ ($1 \leq \gamma \leq 9$), which is used to set the time of rotation to $0.003|10 - x_{\text{image}}|\gamma$ seconds. When $0 \leqq x_{\mathrm{IR}} < \beta_0$, the robot randomly chooses moving backward or turning right to avoid collisions.

B. Using (1+1) Restart-Online Adaptation Algorithm

To achieve a high performance F, K of the swarm, we have to seek good combinations of values for the three parameters α, β, γ. It is natural to borrow the strength of online evolutionary adaptation algorithms, which have proven to be highly effective in multi-robot systems [11], [12]. We use the (1+1) ROAA [7], which allows the robot to search among good controllers without using communication. A genome is represented as a triplet of values for (α, β, γ) and the algorithm basically compares the current champion genome with a challenger genome with some randomness.

The (1+1)-ROAA shares strong links with algorithms from Evolution Strategies (ES) [17], in particular with the (1+1)-ES algorithm. In both cases, a single individual (the child) is evaluated and competes with the best individual found so far (the parent). The child results from a slight (Gaussian) mutation of the parent, and replaces the parent only if it achieves better performance.

The major differences between the original (1+1)-ES algorithm and the (1+1)-ROAA are due to the very nature of the search landscape the algorithm explores, which is both ill-structured, i.e., the same behavior may yield different performance from different starting conditions, and multi-modal, i.e., there is a risk of premature convergence towards good, but not best, solutions. To address these issues, the (1+1)-ROAA endows two mechanisms: the first one is dedicated to mutation parameter updates, which relies on the assumption that a quick increase in performance is a good indication of a good search region, i.e., the mutation is low if the performance increases quickly. Note that this is very different from the traditional (1+1)-ES mutation update scheme, which assumes a convex function optimization. The second mechanisms is the restart process: if the search is stalled, the algorithm is restarted from a different initial condition, which ultimately makes it possible to combine (random) global search with ((1+1)-ROAA) local search. The latter is inspired from a similar restart procedure from a recent ES search algorithm [18].

The pseudo code of the (1+1) ROAA is given in Table II. Line 2 defines the genome of the initial champion ($\mathrm{P}_{\mathrm{Champion}}$) with function SetChampion() and line 3 sets the value of the radius σ to its minimum value σ_{min}. Note that σ is used to determine the area of the proximity in search. The code from line 4 to 28, which evaluates genomes, is iterated τ times.

Lines 5 to 15 are basically executed with probability ReEvaluationRate, as function random() returns a random value from 0.0 to 1.0. Here, $\mathrm{P}_{\mathrm{Champion}}$ is re-evaluated if the number $C_{\mathrm{reevaluation}}$ of evaluations of the champion is smaller than a specified number Reevaluation_Max. Function Recover(P) makes the robot with controller P escape from others, i.e., the robot performs the actions for the situation "preceding robot NOT FOUND" in Table I for a specified time (ζ seconds) to avoid deadlocks and the influence of the previous genome on the next genome. $\mathrm{F}_{\mathrm{Champion}}$ and $\mathrm{F}_{\mathrm{Challenger}}$ represent the fitness values of the champion and that of the challenger, respectively. The fitness value of a specific genome under a specific condition, which includes the controllers of other robots and the positions of the robots, is calculated with function RunAndEval(). RunAndEval() returns the add-sum of the values of function EVALUATE() defined in Table III for each η second during η_0 seconds. The values in Table III were set manually so that the performance of the move in column formation is reflected in the fitness values appropriately. If the number of evaluations of the champion is no smaller than Reevaluation_Max, a new champion is created randomly by the function RandomGenome() and evaluated by RunAndEval().

From line 16 to 27, a challenger genome ($\mathrm{P}_{\mathrm{Challenger}}$) is

TABLE I
CONTROLLER OF A ROBOT

Information from the image sensors	Information from the IR sensors		
	$0 \leqq x_{\text{IR}} < \beta_0$	$\beta_0 \leqq x_{\text{IR}} < \beta$	$\beta \leqq x_{\text{IR}}$
$0 \leqq x_{\text{image}} \leqq \alpha$	move backward or turn right	turn left	
$\alpha < x_{\text{image}} < (19-\alpha)$	move backward or turn right	stop	move forward
$\alpha \leqq x_{\text{image}} \leqq 19$	move backward or turn right	turn right	
preceding robot NOT FOUND	move backward or turn right	move forward	

TABLE III
RETURN VALUE OF FUNCTION EVALUATE()

Information from the image sensor	Information from the IR sensors		
	$0 \leqq x_{\text{IR}} < \beta_0$	$\beta_0 \leqq x_{\text{IR}} \leqq 40$	$40 < x_{\text{IR}}$
$0 \leqq x_{\text{image}} \leqq J$	0	1.0	0.7
$J < x_{\text{image}} < (19-J)$	0	0.4	0
$J \leqq x_{\text{image}} \leqq 19$	0	1.0	0.7
preceding robot NOT FOUND		0	

TABLE II
PSEUDO CODE OF (1+1) RESTART-ONLINE ADAPTATION ALGORITHM FOR EVOLVING A CONTROLLER

```
1  function ONEtoONE_online_ algorithm{
2    P_Champion = SetChampion()
3    σ = σ_min, C_reevaluation ≤ 0
4    for m = 1 to τ do
5      if random() < ReEvaluationRate or m == 1 then
6        if C_reevaluation < Reevaluation_Max then
7          Recover(P_Champion)
8          F_Champion = (F_Champion + RunAndEval(P_Champion))/2
9          C_reevaluation+ = 1
10       else
11         σ = σ_min
12         P_Champion = RandomGenome()
13         F_Champion = RunAndEval(P_Champion)
14         C_reevaluation = 0
15       end if
16     else
17       P_Challenger = P_Champion + N(0, σ)
18       Recover(P_Challenger)
19       F_Challenger = RunAndEval(P_Challenger)
20       if F_Challenger > F_Championthen
21         P_Champion = P_Challenger
22         F_Champion = F_Challenger
23         σ = σ_min
24       else
25         σ = MIN(2.0σ, σ_max)
26       end if
27     end if
28   end for
29 }
```

created in the proximity of the champion genome. $N(0, \Sigma)$ represents a 3-dimensional vector consisting of random values generated based on a Normal distribution with the mean (0, 0, 0) and the covariance matrix Σ, which is a unit matrix multiplied by σ. The challenger becomes a new champion if it outperforms the current champion, otherwise the value of σ is doubled unless it surpasses its maximum value σ_{max}. MIN(a, b) returns the smaller value of a and b.

Whenever the champion is reevaluated, its stored fitness value is partially updated by averaging the old fitness value with the newly computed one. Such an aggregation scheme may overcome the potential negative effect of noisy fitness evaluations in some specific cases, e.g., a reliable genome suffering from a critical starting point [19].

IV. INTEGRATING RULE DISCOVERY

Limiting the combinations of the values of the parameters (α, β, γ) to those resulted in good F, K would improve the overall performance of the swarm during the experiments. Note that the question is to characterize the good performing combinations and is not to discriminate them from the bad ones. Therefore we employ classification rule discovery to identify such promising regions[4].

In the terminology of classification rule discovery, an instance is defined as a triplet of values for (α, β, γ) and its class label. From the viewpoint of evolutionary computing, the triplet of values for (α, β, γ) together form a genome, while the class label is determined by its fitness value. Let t_p represent the time used to evaluate one genome. Robot i has a fixed genome from time $t = (m-1)t_p$ to mt_p and we need a procedure to determine the class label of the genome. For this purpose, we invented $F_{\text{ind}}(i, m)$ and $V_{\text{ind}}(i, m)$, which respectively correspond to F and V for robot i. The class label is good if both $F_{\text{ind}}(i, m)$ and $V_{\text{ind}}(i, m)$ are above average and bad otherwise.

$F_{\text{ind}}(i, m)$ is defined as the ratio of the time the robot i is pursuing its preceding robot based on its genome. The sampling rate used for this measure is determined by the USB camera on the ceiling.

$$F_{\text{ind}}(i, m) = \frac{1}{t_p} \sum_{t=(m-1)t_p}^{mt_p} \mathcal{F}(i, t) \tag{4}$$

Let the order of a robot in a column of k robots be robots $1, 2, \ldots, k$ from the leader, its follower, ..., to the last one, respectively. $V_{\text{ind}}(i, m)$ must not be simply the average velocity of the robots because the larger the order of the robot in

[4]Classification methods such as decision tree learning are for discriminating different class and are thus not used in our framework.

its column is, the slower its velocity tends to be, mainly due to stacking. The simple choice would mislead us in evaluating the degree of activity of a genome considerably. We adopt the average velocity $v_{\rm I}(i,t)$ of the leader of robot i for computing $V_{\rm ind}(i,m)$.

$$V_{\rm ind}(i,m) = \frac{1}{F_{\rm ind}(i,m)} \sum_{t=(m-1)t_p}^{mt_p} v_I(i,t)\mathcal{F}(i,t) \qquad (5)$$

Preliminary experiments show that the ratio of instances of the good class is about 20%, whereas an instance may be considered bad depending on the combinations of the controllers of the robots and their positions. Thus, we restrict our attention to rules that predict good instances. For a rule "If $Y \rightarrow$ class = good", where Y represents the premise, we follow the widely used approach of specifying minimum thresholds values for the support $P(\text{class} = \text{good}, Y)$ and the confidence $P(\text{class} = \text{good}|Y)$. Here Y is either a genome or a generalization of a genome, i.e., a conjunction of 2 or 3 atoms or a single atom, where an atom is (parameter = value).

V. Experimental Evaluation

A. Robot

We used a robot kit called Robo Designer, which is commercially available from Japan Robotech. Its size is approximately 19.0 cm (length) $\times$ 18.0 cm (width) $\times$ 15.5 cm (height). The speed of robots vary from 9.1cm/s to 16.5cm/s. We use SHARP GP2Y0A21YK as the IR sensor, which is able to measure distances from 5 cm to 50 cm. The sensor reading for an obstacle closer than 5 cm is highly unreliable while the sensor reading for an obstacle located farther than 50 cm is equivalent to that for 50 cm.

For the image sensor, we use CMOS EYE of Asakusa Giken, with a maximum transmission speed 460,800bps, a perception angle 52 degrees, and a resolution of a color image ranging from 20 pixels $\times$ 15 pixels to 640 pixels $\times$ 480 pixels. We adopted the resolution of 20 pixels $\times$ 15 pixels so as to store the RGB and HSV formats of an image on the MPU, which has a RAM of 8KB[5]

We use 3-color LED of AVAGO Technology with a maximum voltage 3.6 V and a maximum amperage 350 mA. The MPU we use is Arduino MEGA, with a maximum transmission speed is 115,200 bps, with a 128 KB flash memory and a 8KB RAM, and its clock speed is 16 MHz.

Initial position of the five robots are placed in a column formation. Coordinates for the small field are (39, 30), (77, 30), (124, 30), (167, 30), and (210, 30); while those for the large field are (39, 85), (77, 85), (124, 85), (167, 85), and (210, 85). We set $\beta_0 = 15$, $\zeta = 2$, $\sigma_{\rm min} = 0.3$, and $\sigma_{\rm max} = 0.7$. In updating the values of α and γ, we use 2.5σ instead of σ for speed up.

[5]Since H, S, and V take values from 0-359, 0-255, 0-255, we use 2, 1, and 1 bytes for them, respectively. Thus we use $15 * 20 * 5 = 1,200$ bytes for an image while we need 4,800 bytes for an image of 30 pixels $\times$ 40 pixels. The RAM can store an image of the latter size but we adopted the former one for speed up.

Fig. 3. Field (left) and the USB camera on the ceiling (right) of the environment of the experiments. For the field, the top leftmost corner is the origin; and x and y axes are along the thermocol blocks and the wall, respectively.

B. Extraction of Log Data from Video

The robots move in an indoor field of size 195 cm $\times$ 255 cm, or of size 250 cm $\times$ 480 cm, as shown in Figure 3. It is surrounded by walls and thermocol blocks and lighted by fluorescent lamps. Note that noise is less problematic in this field than in an outdoor setting with sunlight. We used $n = 5$ robots due to the size of the field and the ease of maintenance.

The video has 15 frames per second and of size 640 pixels $\times$ 480 pixels. The height of the thermocol blocks is 19 cm and we place a black plastic plate of height 60 cm behind the blocks to shut out illumination from the next office. Note that even with this measure, the readings of the IR sensor depend on the perimeters due to their different heights, materials, and colors. We have also found that the illumination varies depending on the location in the field. Furniture such as tables and chairs located outside of the field turned out to be a source of noise. The five robots are covered with sheets of papers colored blue, light blue, green, orange, and pink for identification.

The extension for searching $\mathbf{s}_i(t)$ is set to 3 cm. From the video analysis, we assume as sight range $\mathbf{A}_i(t)$ of 66 degrees, i.e., $\varphi = 33$ degrees in this case, and we adopt $l_1 = 5$ cm and $l_2 = 50$ cm. For calculating V in Eq. (3), we use a sampling rate of 1 second for a robust estimation. We set $\eta = 0.3$ and $\eta_0 = 60$.

C. Results of Experiments

The number of possible combinations of the parameters α, β, γ of a controller is 2200 for our problem. Note that a specific combination typically exhibits different values for F and V, as they depend of the combination of the controllers of the five robots and their positions. This characteristic is challenging and justifies our use of the evolutionary algorithm and rule discovery.

For rule discovery, we use Jrip and NNge in the Weka package [20] with their default settings. Jrip is an extension of Repeated Incremental Pruning to Produce Error Reduction (RIPPER) [21]. NNge, which stands for Non-nested generalised exemplars, discovers rules based on the nearest-neighbor approach, merging examples to find hyperrectangles in attribute space, i.e., conjunction rules [22]. The minimum support and confidence thresholds are set to 2/720 = 0.00277 and 0.8, respectively, for both algorithms.

TABLE IV
RULES WHICH ARE DISCOVERED FROM THE LOG FOR THE SMALL FIELD

Rule name	parameter setting			sup.	conf.
	α	β	γ		
R_s00	$\alpha = 3$	$\beta = 20$	$6 \leqq \gamma \leqq 8$	5	1.00
R_s01	$\alpha = 4$	$\beta = 19$	$1 \leqq \gamma \leqq 4$	3	1.00
R_s02	$4 \leqq \alpha \leqq 5$	$24 \leqq \beta \leqq 26$	$6 \leqq \gamma \leqq 7$	4	1.00
R_s03	$4 \leqq \alpha \leqq 6$	$\beta = 20$	$\gamma = 5$	5	0.80
R_s04	$5 \leqq \alpha \leqq 8$	$26 \leqq \beta \leqq 27$	$\gamma = 4$	2	1.00
R_s05	$7 \leqq \alpha \leqq 8$	$20 \leqq \beta \leqq 24$	$5 \leqq \gamma \leqq 6$	5	0.80

TABLE V
RULES WHICH ARE DISCOVERED FROM THE LOG FOR THE LARGE FIELD

Rule name	parameter setting			sup.	conf.
	α	β	γ		
R_l00	$3 \leqq \alpha \leqq 4$	$22 \leqq \beta \leqq 23$	$\gamma = 6$	3	1.00
R_l01	$\alpha = 4$	$26 \leqq \beta \leqq 27$	$6 \leqq \gamma \leqq 7$	3	1.00
R_l02	$\alpha = 4$	$24 \leqq \beta \leqq 29$	$\gamma = 3$	2	1.00
R_l03	$4 \leqq \alpha \leqq 6$	$22 \leqq \beta \leqq 23$	$\gamma = 5$	2	1.00
R_l04	$\alpha = 5$	$26 \leqq \beta \leqq 28$	$\gamma = 8$	2	1.00
R_l05	$6 \leqq \alpha \leqq 7$	$28 \leqq \beta \leqq 30$	$\gamma = 5$	2	1.00
R_l06	$6 \leqq \alpha \leqq 8$	$\beta = 25$	$6 \leqq \gamma \leqq 7$	2	1.00

For the (1+1) ROAA, we set $\tau = 26$ and ReEvaluationRate $= 0.3$. We rather wanted to set $\tau = 26 * 6$ but this choice resulted in evolutions of the robots that were not well synchronized when t becomes large. So, we instead applied the evolutionary algorithm in Table II six times. Thus we obtain $6 \times 5 \times 26 = 720$ instances in a series of experiments. Since a controller is evaluated in about 60 seconds, a series of experiments lasts about 2 hours and half.

The discovered rules are shown in Table IV and Table V for the small and large fields, respectively. They are all discovered by NNge as Jrip, which we suspect as being conservative in discovery, returned no result.

Table VI shows the performance of controllers without and with discovered rules. In the table, "s" and "l" represent the small and large fields, respectively. "None" and "all" represent using no rule and all discovered rules, respectively. We also evaluated the best combinations of parameter values that show highest values for $V_{\mathrm{ind}}(i, m)$ among the controllers evaluated at least twice, which are marked with "best".

The ratios of good instances for the small field are 22%, 25%, and 30% for without/with rules and best, respectively.

TABLE VI
PERFORMANCE OF CONTROLLERS WITHOUT AND WITH DISCOVERED RULES

Controller name	used rules or genome	results	
		F	V
Controller-none$_s$	none	0.44	6.1
Controller-none18$_s$	none	0.45	6.1
Controller-none18$_s$-best	$\alpha = 4$, $\beta = 20$, $\gamma = 5$	0.62	6.5
Controller-R$_s$all	R$_s$00-05	0.51	6.8
Controller-R$_s$all-best	$\alpha = 8$, $\beta = 21$, $\gamma = 6$	0.41	8.6
Controller-none$_l$	none	0.35	7.1
Controller-none18$_l$	none	0.38	7.7
Controller-none18$_l$-best	$\alpha = 9$, $\beta = 21$, $\gamma = 9$	0.20	9.7
Controller-R$_l$all	R$_l$00-06	0.49	8.8
Controller-R$_l$all-best	$\alpha = 4$, $\beta = 19$, $\gamma = 6$	0.47	9.1

TABLE VII
PERFORMANCE OF CONTROLLERS AND WITH DISCOVERED GROUPS FROM RULES

Controller name	used rules or genome	results	
		F	V
Controller-G$_s$	R$_s$00, 01, 03, 05	0.59	7.0
Controller-G$_s$-best	$\alpha = 7$, $\beta = 23$, $\gamma = 6$	0.54	8.0
Controller-R$_s$02	R$_s$02	0.65	7.4
Controller-R$_s$02-best	$\alpha = 5$, $\beta = 24$, $\gamma = 7$	0.53	7.4
Controller-G$_l$0	R$_l$00, 03	0.59	8.8
Controller-G$_l$0-best	$\alpha = 6$, $\beta = 23$, $\gamma = 5$	0.68	9.4
Controller-G$_l$1	R$_l$01, 04	0.50	8.6
Controller-G$_l$1-best	$\alpha = 4$, $\beta = 26$, $\gamma = 7$	0.56	9.0

Those for the large field are 30%, 34%, and 39%. In the Table, Controller-none and Controller-none18 represent the first series of experiments using ROAA without discovered rules for 6 and 18 controllers. We have checked the standard deviations of F and V for Controller-none18. For the small field, they are 1.9311 and 2.325, respectively, and for the large field: 1.6526 and 2.984. Further analyses of each discovered rule are given in the next section.

D. Using Similar Rules Only

We obtain groups of rules for small and large fields by iteratively grouping a pair of rules if the closest instances of their coverages satisfy $\alpha^2+\beta^2+\gamma^2 < 2$, which we adopt as the criteria of similar rules. Then, we tested (1+1) ROAA using similar rules only to investigate the impact of the discovered rules to our framework. For this purpose, we select the two largest groups in terms of the number of instances a group covers for each field.

For the small field, we obtained three groups: G_s (R$_s$00, 01, 03, 05), R$_s$02, and R$_s$04, which cover 30, 12, and 8 instances, respectively. G_s may be interpreted as rules with relatively short distances to the preceding robot while R$_s$02 imposes a longer distance. We performed experiments using rules only in G_s and using only R$_s$02.

For the large field, we obtained five groups: G_l0 (R$_l$00,03), G_l1 (R$_l$01,04), R$_l$02, R$_s$05, and R$_s$06, which cover 10, 7, 6, 6, and 6 instances, respectively. G_l0 may be interpreted as imposing a short distance and medium values for the velocities of turns, whereas G_l1 imposes a long distance and large values for the velocities of turns. We did experiments using rules only for G_l0 or G_l1.

We have analyzed these results and compared them with those of the previous section. For the small field, the best rule in terms of F is Controller-none18$_s$-best, which was identified without relying on the rule discovery algorithm. We attribute the reason for the superiority of Controller-none18$_s$-best to the fact that a robot has a high chance to see another robot in the small field so many genomes exhibit good performance in terms of F. This explanation is supported by the fact that Controller-G$_s$, Controller-R$_s$, and Controller-none18$_s$ show a similar performance.

For the large field, the best rule in terms of F is Controller-G$_l$0-best, which was identified thanks to the rule discovery algorithm. Since a robot has a low chance to find another

robot in the large field, it is necessary to keep good genomes. Otherwise, the robots are widely distributed, which renders column formation difficult. This explanation is supported by the fact that each of Controller-G$_l$0 and Controller-G$_l$1 exhibits better performance than Controller-R$_l$all.

In terms of the parameter values, Controller-none18$_s$-best, Controller-R$_s$02, and Controller-G$_l$0-best, which exhibit good F, satisfy $4 \leq \alpha \leq 6$, $\beta = 20$ or $23 \leq \beta \leq 26$, and $5 \leq \gamma \leq 7$. From these conditions, we can say that the velocity of turns are moderate.

VI. Conclusions and Future Work

In this paper, we proposed a framework for designing autonomous robots that move in column formation based on rule discovery. Each robot adopts (1+1) ROAA for searching good combinations of parameter values. We applied a rule discovery algorithm to the log data of experiments and identified rules which describe promising combinations of the parameter values. In later experiments, the robots perform the search among the promising combinations in order to have a superior performance. Experimental results using five physical robots are encouraging and justify our approach.

In future work, the discovered rules may be used for performing finer search in the promising subregions of the instance space. Note that in any non-trivial optimization process, coarse-to-fine search is one of the most effective ways for speed up, e.g., [23]. In the finer search, it is mandatory to investigate the minimum number of runs for each controller to obtain reliable estimates of the performance measures. Strictly speaking, the evolution should take place at the group level [24] and not at the individual level. We plan to use the WIFI or Bluetooth units for allowing the robots to exchange their genomes to enable such a kind of evolution. It is practical to use communication among robots as a way to resolve deadlocks.

Employing more sophisticated machine learning and data mining methods for more complex tasks than column formation would be highly beneficial in designing self-adapted multi-robot systems. For instance, discovering combinations of conditions which result in different outcomes [25] would give deeper insights to the human designer. In the long run, on the other hand, we believe that on-board optimization of the controller provides new challenges to machine learning and data mining, eventually opening new research avenues in these fields.

Acknowledgment

A part of this research was supported by Strategic International Cooperative Program funded by Japan Science and Technology Agency (JST) on the Japanese side and Agence Nationale de Recherches (ANR) on the French side. On the French side, this work was also funded by the European Union FET Proactive Initiative: Pervasive Adaptation funding the Symbrion project under grant agreement 216342.

References

[1] M. Schwager, J. McLurkin, and D. Rus, "Distributed Coverage Control with Sensory Feedback for Networked Robots," in *Proc. Robotics: Science and Systems*, August 2006.

[2] N. Jakobi, P. Husbands, and I. Harvey, "Noise and The Reality Gap: The Use of Simulation in Evolutionary Robotics," in *Proc. ECAL 1995*, 1995, pp. 704–720.

[3] H. Lipson, "Evolutionary Robotics: Emergence of Communication," *Current Biology*, vol. 17, no. 9, pp. R330–R332, 2007.

[4] A. Saxena and A. Saad, "Evolving an Artificial Neural Network Classifier for Condition Monitoring of Rotating Mechanical Systems," *Applied Soft Computing*, vol. 7, no. 1, pp. 441–454, 2007.

[5] A. Kouno, S. Takano, and E. Suzuki, "Constructing Low-cost Swarm Robots that March in Columnn Formation," in *Swarm Ingeligence (ANTS 2010), LNCS 6234*, 2010, pp. 556–557.

[6] N. Bredeche, E. Haasdijk, and A. E. Eiben, "On-line, On-board Evolution of Robot Controllers," in *Artificial Evolution (EA'09) – Selected Papers*. LNCS 5975, Springer-Verlag, 2010, pp. 110–121.

[7] J.-M. Montanier and N. Bredeche, "Embedded Evolutionary Robotics: The (1+1)-Restart-Online Adaptation Algorithm," *Exploring New Horizons in Evolutionary Design of Robots*, pp. 37 – 43, 2009.

[8] D. J. Hand, P. Smyth, and H. Mannila, *Principles of Data Mining*. Cambridge, Mass.: MIT Press, 2001.

[9] M. J. Matarić, "Reinforcement Learning in the Multi-Robot Domain," *Autonomous Robots*, vol. 4, no. 1, pp. 73 – 83, 1997.

[10] P. Stone, "Learning and Multiagent Reasoning for Autonomous Agents," in *Proc. IJCAI 2007*, 2007.

[11] V. Trianni, S. Nolfi, and M. Dorigo, "Evolution, Self-organization and Swarm Robotics," in *Swarm Intelligence: Introduction and Applications*. Springer-Verlag, 2008, pp. 163 – 191.

[12] R. A. Watson, S. G. Ficici, and J. B. Pollack, "Embodied Evolution: Distributing an Evolutionary Algorithm in a Population of Robots," *Robotics and Autonomous Systems*, vol. 39, no. 1, pp. 1 – 18, 2002.

[13] A. F. T. Winfield, "Foraging Robots," in *Encyclopedia of Complexity and System Science*, R. Meyers, Ed. Springer, 2009, pp. 3682–3700.

[14] S. Hauert, C. Z. J, and D. Floreano, "Evolved Swarming without Positioning Information: an Application in Aerial Communication Relay," *Autonomous Robots*, vol. 26, no. 1, pp. 21–32, 2009.

[15] I. Navarro and F. Matia, "A Proposal of a Set of Metrics for Collective Movement of Robots," in *Proc. Workshop on Good Experimental Methodology in Robotics*, 2009.

[16] J. Gautrais, C. Jost, and G. Theraulaz, "Key Behavioural Factors in a Self-organised Fish School Model," *Annales Zoologici Fennici*, vol. 45, no. 5, pp. 415 – 428, 2008.

[17] H.-G. Beyer and H.-P. Schwefel, "Evolution Strategies: A Comprehensive Introduction," *Journal Natural Computing*, vol. 1, no. 1, 2002.

[18] A. Auger and N. Hansen, "A Restart CMA Evolution Strategy With Increasing Population Size," in *Proc. CEC 2005*, 2005, pp. 1769 – 1776.

[19] A. E. Eiben, E. Haasdijk, and N. Bredeche, "Embodied, On-line, On-board Evolution for Autonomous Robotics," in *Symbiotic Multi-Robot Organisms: Reliability, Adaptability, Evolution*, ser. Cognitive Systems Monographs. Springer-Verlag, 2010, vol. 7, pp. 361–382.

[20] I. H. Witten and E. Frank, *Data Mining, Second Edition*. Morgan Kaufmann, 2005.

[21] W. W. Cohen, "Fast Effective Rule Induction," in *Proc. ICML 1995*, 1995, pp. 115–123.

[22] B. Martin, "Instance-based learning: Nearest neighbour with generalisation," Computer Science, University of Waikato, Hamilton, Working Paper Series 95/18, 1995.

[23] H. Iba, T. Higuchi, H. de Garis, and T. Sato, "Evolutionary Learning Strategy using Bug-Based Search," in *Proc. IJCAI 1993*, 1993, pp. 960 – 966.

[24] N. Bredeche, J.-M. Montanier, W. Liu, and A. F. T. Winfield, "Environment-driven Distributed Evolutionary Adaptation in a Population of Autonomous Robotic Agents," *Mathematical and Computer Modelling of Dynamical Systems*, 2011.

[25] E. Suzuki, "Autonomous discovery of reliable exception rules," in *Proc. of KDD-97*, 1997, pp. 259–262.