Modular Automatic Design of Collective Behaviors for Robots Endowed with Local Communication Capabilities

K. Hasselmann and M. Birattari
Modular automatic design of collective behaviors for robots endowed with local communication capabilities

Ken Hasselmann · Mauro Birattari

Abstract We investigate the automatic design of communication in swarm robotics through two studies. We first introduce Gianduja, an automatic design method that generates collective behaviors for robot swarms in which individuals can locally exchange a message whose semantics is not a priori fixed. It is the automatic design process that, on a per-mission basis, defines the conditions under which the message is sent and the effect that it has on the receiving peers. Then, we extend Gianduja to Gianduja2 and Gianduja3, which target robots that can exchange multiple distinct messages. Also in this case, the semantics of the messages is automatically defined on a per-mission basis by the design process. Gianduja and its variants are based on Chocolate, which does not provide any support for local communication. In the paper, we compare Gianduja and its variants with a standard neuro-evolutionary approach. We consider a total of six different swarm robotics missions. We present results based on simulation and tests performed with 20 e-puck robots. Results show that, typically, Gianduja and its variants are able to associate a meaningful semantics to messages.

1 Introduction

Communication can have a major impact on the performance of groups of robots (Kirby 2002). This is particularly true in swarm robotics, where robots are deemed to cooperate to effectively perform a mission that an individual would not be able to perform alone (Dorigo et al. 2014). In general, designing collective behavior for robot swarms is a challenging endeavor (Brambilla et al. 2013). Including effective communication mechanisms in the design makes it even more challenging (Balch 2004).
Automatic methods are a promising way for designing robot swarms (Francesca and Birattari 2016). In automatic design, the design problem is reformulated as an optimization problem in which design choices are the parameters to be optimized. An optimization algorithm searches the space of the possible solutions to maximize a mission-specific performance metric. Most of the advances in the automatic design of robot swarms belong in the neuro-evolutionary approach; robots are controlled by an artificial neural network that is trained via an evolutionary algorithm—for example, see Jakobi et al. (1995); Urzelai and Floreano (2000); Nolfi and Floreano (2000); Trianni (2008); Gauci et al. (2014); Silva et al. (2015). Besides neuro-evolution, other approaches have been proposed. For example, Francesca et al. (2014a) used probabilistic finite state machines as a control architecture for the individual robots, while Jones et al. (2016) and Kuckling et al. (2018) used behavior trees.

In this paper, we study the automatic design of collective behaviors for robot swarms that are capable of local communication. We consider robots that can locally exchange messages. Our goal is to define and study methods that design collective behaviors for these robots and automatically define a semantics for the messages they can exchange. We propose Gianduja, an automatic design method that extends Chocolate (Francesca et al. 2015) by adding messaging capabilities to the robots—Gianduja and Chocolate are described in Section 3 and following. In Gianduja, robots can exchange a single bit of information, which amounts to broadcasting a message or not. We consider also two variants of Gianduja: Gianduja2 and Gianduja3, in which robots can exchange two and three bits of information, respectively—details are provided in Section 3 and following. We first study Gianduja on three missions that a robot swarm can perform relying on exchanging one bit of information. Then we study Gianduja2 and Gianduja3 on more complex missions in which a one-bit communication is not sufficiently informative to allow the coordination of the swarm.

Using a neuro-evolutionary approach, Quinn (2001) has already shown that it is possible to assign an implicit semantics to a message via an automatic design process. Nonetheless, his result is only marginally relevant to our research as it concerns only a pair of robots rather than a swarm. Moreover, the result was obtained on a single specific mission and does not immediately generalize to other missions. The study was performed in simulation only. In this paper, we automatically design collective behaviors based on communication for six different missions and we test them on a swarm of 20 e-puck robots. For all missions considered, the automatic design method was able to assign an implicit semantics to the messages that robots exchange.

It should be noted that the research we present in this paper focuses on pure off-line, automatic design of control software for robot swarms (Francesca and Birattari 2016): for a given mission, control software is automatically designed in simulation, ported to robots, and assessed. A mission is specified via a performance measure, an arrangement of the environment, and the deployment area of the robots. The tenets of our research are that (i) automatic design
methods must be mission agnostic and should be able to address a whole
class of missions; (ii) once a mission is specified, human intervention is not
provided for in any phase of the design process. Namely, we exclude the case
in which design methods are conceived for or manually adapted to a specific
mission—for example, by tuning parameters of the optimization algorithm
and/or of the control architecture. We also exclude the case in which human
designers can inspect the behavior of an automatically designed swarm either
in simulation or in reality and, on the basis of their observations, are allowed
to modify elements of the automatic design process and iterate it at will,
until a satisfactory behavior is obtained. In particular, human designers are
not allowed to use the insight gained through inspection to: modify the design
method (optimization algorithm, architecture, sensor pre-filtering, etc.), adapt
simulation models, add/remove terms to the objective function to steer the
design process as wished.

2 Related Work

The emergence of signaling, syntax, and language in artificial agents has been
widely studied from the perspectives of language sciences and engineering—
for example, Steels (1998); Billard and Dautenhahn (1999); Cangelosi (2001);
Loula et al. (2010). The literature is extensive when it comes to multi-robot
systems that use a form of communication. However interesting, many of these
studies are outside the scope of this paper because they concern systems
in which communication is \textit{a priori} designed—for example, Balch and Arkin
(1994); Cao et al. (1997); Jones and Mataric (2004); Fong and Nourbakhsh
(2009); Balch (2004); Werfel et al. (2014). We focus here on studies that in-
volve the automatic design of communication in swarm robotics. Most existing
studies on communication in swarm robotics were conducted in the framework
of neuro-evolutionary robotics (Nolfi and Floreano 2000; Trianni et al. 2004;
Wischmann and Pasemann 2006; Trianni 2008; Wischmann et al. 2012; Tri-
ni 2014). Quinn (2001) and Quinn et al. (2003) studied for the first time the
emergence of communication between agents. In their study, robots developed
communicative behaviors by detecting certain motion patterns of their peers,
without the use of a dedicated communication device. The robots established
a social interaction and assumed leader and follower roles. In Nolfi (2005) and
Marocco and Nolfi (2006), a group of robots evolved to solve a collective nav-
igation problem. Robots were controlled by a neural network and could use
four different signals. The evolutionary process produced behaviors that make
an effective use of communication without directly rewarding communicative
behaviors. The system was tested in simulation and on a swarm of four robots.
Floreano et al. (2007) also studied the evolution of communicative behaviors
using neuro-evolutionary robotics. In their study, the robots used visual signals
to communicate the location of a food source. In their experiments conducted
with physical robots, communication increased the performance of the swarm
with respect to a non-communicative swarm. In Ampatzis et al. (2008), two
robots had to recognize features of their environment and react accordingly. The robots were controlled by neural networks and could use their on-board speakers and microphones to communicate using acoustic signals. During the evolutionary process, communication emerged as a result of the performance improvement it allowed, even though it was not directly rewarded. The authors tested the performance of the robots both in simulation and reality with two s-bot robots (Mondada et al. 2003). The origin of communication from a neuro-evolutionary perspective was studied by Tuci (2009). The same two robots considered in the previous study had to perform a categorization task using communication based on acoustic signals. The evolutionary process produced behaviors that use communication to improve performance although communication was not explicitly rewarded. Experiments were conducted in simulation only. Communication was also studied by Wischmann et al. (2012), who compared different communication strategies evolved from the same initial population and highlighted the trade-off between communication efficiency and robustness. Experiments were run in simulation only.

The research we present in this paper is related to those of Ampatzis et al. (2008) and Tuci (2009). Indeed, our goal is to define an automatic design process in which messages that do not have an a priori semantics are effectively used to communicate. Messages are associated with a semantics by the design process, on a per-mission basis.

3 Gianduja

Gianduja and all its variants that we will present in the following are based on Chocolate (Francesca et al. 2015). As Chocolate, they belong to the AutoMoDe class of methods originally defined by Francesca et al. (2014b). These methods automatically generate control software by assembling predefined, mission-agnostic software modules. Like Chocolate, Gianduja produces control software for the e-puck platform (Mondada et al. 2009), extended with three hardware modules:¹ the Overo Gumstix, the ground sensor, and the range and bearing. The e-puck is a circular two-wheeled robot, whose diameter is about 70 mm. It has 8 IR transceivers, positioned all around its body, that double as light and proximity sensors. The Overo Gumstix module is a single-board computer that allows the e-puck to run Linux. The ground sensor module allows the e-puck to perceive the color of the floor. The range-and-bearing module (Gutiérrez et al. 2009) is an infrared communication device for local sensing and messaging. It operates by broadcasting every 100 ms a 2-byte ping: one byte encodes the sender ID and the other an arbitrary payload. The ping can be received by robots within a range of about 0.7 m from the sender. A robot that receives a ping is able to estimate the relative position of the sender in polar coordinates—range and bearing, hence the name of the device.

¹ see http://www.e-puck.org
sembled using pre-existing, mission-independent modules. Modules are either behaviors to be used as states of the state machine or conditions to be associated with its edges. Conditions determine whether a transition should happen or not. Modules may have tunable parameters that modify their functioning. The topology of the probabilistic finite state machine, the behaviors and the conditions to be included, and the value of their parameters are determined by an optimization algorithm that maximize a mission-specific performance measure. The optimization algorithm adopted in Gianduja, as well as in Chocolate, is iterated F-race (Balaprakash et al. 2007; Birattari et al. 2010). Within the optimization process, simulations are performed using ARGoS3—details are provided in Section 4.1.1.

The only element in which Gianduja differs from Chocolate is that in Gianduja the modules have parameters that control local communication between robots. Each robot can locally broadcast messages and receive those broadcast by neighboring peers. Messages are a priori pure signifiers: they are not associated to any semantics. It is the optimization algorithm that, on a per-mission basis, gives a meaning to messages, that is, associates a semantics to signifiers. In the context of this research, giving a meaning to a message means to define (i) the conditions under which the sender broadcasts the message and (ii) the effects that the message has on the recipient’s behavior.

In study A, we focus on the automatic design of collective behaviors for robots that locally broadcast one bit of information using the range-and-bearing module. In practice, robots communicate by setting a one-bit flag of the ping’s payload. A robot that perceives a neighboring peer via the range-and-bearing module—that is, that receives its ping—will read the flag: if it is set, the neighboring peer is broadcasting the message; otherwise, it is not.

In study B, we generalize study A to robots that locally broadcast two (or three) bits of information via the range-and-bearing module. In this case, a robot communicates by writing two (or three) bits of the ping’s payload. As a result, robots broadcast/react to three (or seven) distinct messages—that is, \(2^2 - 1\) (or \(2^3 - 1\)), where the \(-1\) accounts for the case in which a robot does not broadcast any message.

Detailed descriptions of Chocolate, of Gianduja, and of a number of its variants are given in Sections 4.1 and 5.1.

4 Study A: One bit of information.

In this section, we study the automatic design of behaviors for robots that exchange one bit of information. In Section 4.1, we present the methods under analysis; in Section 4.2, we describe the experimental setting; in Section 4.3, we report and discuss the results of the study.

4.1 Design methods

In this study A, we consider seven design methods.
Table 1: Reference model RM 1.1.

<table>
<thead>
<tr>
<th>Input Value Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>prox (i \in {1, \ldots, 8})</td>
<td>reading of proximity sensor (i)</td>
</tr>
<tr>
<td>light (i \in {1, \ldots, 8})</td>
<td>reading of light sensor (i)</td>
</tr>
<tr>
<td>gnd (j \in {1, 2, 3})</td>
<td>reading of ground sensor (j)</td>
</tr>
<tr>
<td>(n)</td>
<td>number of neighboring robots perceived</td>
</tr>
<tr>
<td>(V)</td>
<td>their relative aggregate position</td>
</tr>
</tbody>
</table>

Output Value Description

<table>
<thead>
<tr>
<th>Value Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(v_k \in {l, r})</td>
<td>target linear wheel velocity (m/s)</td>
</tr>
</tbody>
</table>

Period of the control cycle: 100 ms

The vector \(V\) is defined as:

\[
V = \begin{cases} 
\sum_{m=1}^{n} (\frac{1}{1+r_m}, \angle b_m), & \text{if robots are perceived;} \\
(1, \angle 0), & \text{otherwise.}
\end{cases}
\]

where \(r_m\) and \(\angle b_m\) are the range and bearing of neighbor \(m\), respectively.

4.1.1 Chocolate

Chocolate (Francesca et al. 2015) generates probabilistic finite state machines by assembling parametric modules—behaviors and conditions—that exploit the capabilities of the e-puck platform, as formally defined by the reference model RM 1.1. For convenience of the reader, we report RM 1.1 in Table 1; further details are available in Hasselmann et al. (2018). It is important to notice that Chocolate’s modules are strictly part of the definition of Chocolate itself. They were defined once and for all in a mission-agnostic way (Francesca et al. 2014b). They are not supposed to be modified or integrated by other modules when Chocolate is applied to a specific mission. Yet, these modules have parameters that impact their functioning and that are automatically fine-tuned on a per-mission basis. In Chocolate, low-level behaviors are:

- exploration: the robot performs a random walk, while avoiding obstacles;
- stop: the robot stands still;
- phototaxis: the robot goes towards the light source, if perceived;
- anti-phototaxis: the robot goes in the opposite direction;
- attraction-to-neighbors: the robot goes towards its neighboring peers;
- repulsion-from-neighbors: the robot goes in the opposite direction.

Conditions are:

- black-floor: change state if floor is black;
- white-floor: change if it is white;
- gray-floor: change if it is gray;
- neighbor-count: change if sufficiently many neighboring peers are perceived;
- inverted-neighbor-count: change if they are sufficiently few;
fixed-probability: change state with a fixed probability.

For details on these modules and their tunable parameters, we refer the reader to the original publication in which the modules were introduced (Francesca et al. 2014b).

The topology of the probabilistic finite state machine, the modules to be included and their parameters are defined by an optimization process. The space of the probabilistic finite state machines that Chocolate can possibly generate is constrained to those comprising at most four states having each at most four outgoing edges. As an optimization algorithm, Chocolate uses the implementation of iterated F-race provided by the R package irace (López-Ibáñez et al. 2016), with its default parameters. Iterated F-race is based on F-race (Birattari et al. 2002), a racing procedure (Maron and Moore 1997) originally proposed for the automatic configuration of stochastic optimization algorithms and metaheuristics (Hoos and Stützle 2004; Gendreau and Potvin 2019). In F-race, a set of candidate solutions are randomly sampled and then sequentially evaluated, over a set of test cases, to eventually select the most suitable one. Along the sequential evaluation of candidate solutions, a Friedman test is repeatedly performed to identify candidate solutions that perform significantly worse than at least another one. These solutions are discarded so that the evaluation can focus on the best ones. The algorithm terminates when only one candidate solution remains or when a predefined budget of evaluations is depleted. Iterated F-race consists of multiple iterations of F-race. After the first iteration, each subsequent one operates on a set of candidate solutions that are sampled around those that the previous iteration selected as the best ones. The algorithm terminates when a predefined budget of evaluations is depleted. Within the optimization process, simulations are performed using the ARGoS3 simulator (Pinciroli et al. 2012), version beta 48, together with the argos3-epuck library (Garattoni et al. 2015). ARGoS3 is a modular multi-physics robot simulator specifically conceived to simulate robot swarms. Chocolate uses ARGoS3’s 2D dynamic physics engine to simulate the robots and the environment. The argos3-epuck library provides low-level implementations of the sensors and actuators of the e-puck robot with fine control on noise levels for all actuators and sensors. ARGoS3 and the argos3-epuck library inject a realistic level of sensor and actuator noise in all simulations as suggested by Miglino et al. (1995) as a good practice for reducing the impact of the reality gap.

As Chocolate is unable to produce behaviors that leverage communication, it will likely fail to be effective on the missions we consider in the following of the paper. We include it in the analysis to act as a baseline, so as to appraise the importance of communication.

4.1.2 EvoCom

EvoCom is an automatic design method that extends EvoStick (Francesca et al. 2014a) by adding communication capabilities. EvoStick is a standard neuro-evolutionary robotics method. It was used in Francesca et al. (2014a) as a
Table 2: Reference model RM 2. Novelties are highlighted with respect to RM 1.1. They concern the ability to send and react to one bit of information. $V_b$ is computed in the same way as $V$—see Table 1—by restricting to broadcasting neighboring robots.

<table>
<thead>
<tr>
<th>Input</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{prox}_{i \in {1, \ldots, 8}}$</td>
<td>$[0,1]$</td>
<td>reading of proximity sensor $i$</td>
</tr>
<tr>
<td>$\text{light}_{i \in {1, \ldots, 8}}$</td>
<td>$[0,1]$</td>
<td>reading of light sensor $i$</td>
</tr>
<tr>
<td>$\text{gnd}_{j \in {1,2,3}}$</td>
<td>{black, gray, white}</td>
<td>reading of ground sensor $j$</td>
</tr>
<tr>
<td>$n$</td>
<td>$[0,20]$</td>
<td>number of neighboring robots perceived</td>
</tr>
<tr>
<td>$V$</td>
<td>$(0.5, 20), [0, 2\pi]$</td>
<td>their relative aggregate position</td>
</tr>
<tr>
<td>$b$</td>
<td>$[0,20]$</td>
<td>number of messaging neighbors perceived</td>
</tr>
<tr>
<td>$V_b$</td>
<td>$(0.5, 20), [0, 2\pi]$</td>
<td>their relative aggregate position</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_{k \in {l,r}}$</td>
<td>$[-0.12, 0.12]$</td>
<td>target linear wheel velocity (m/s)</td>
</tr>
<tr>
<td>$s$</td>
<td>{on, off}</td>
<td>broadcast state</td>
</tr>
</tbody>
</table>

Period of the control cycle: 100 ms
scalar projection of $V_b$ onto the aforementioned four unit vectors. The two output nodes encoding wheel speed range in $[-0.12, 0.12]$ m/s. The one devoted to the message takes a binary value: 1 if the message should be broadcast, 0 otherwise. The activation of the output neurons is computed as the weighted sum of all input units plus a bias term, filtered through a logistic function. All synaptic weights are real values in the range $[-5, 5]$ and are optimized with an evolutionary algorithm. The initial population comprises 100 randomly generated individuals. At each iteration, each individual is evaluated 10 times in simulation. A new population is obtained via elitism and mutation: the best 20 individuals are kept unmodified, while the other 80 are obtained by mutation of the 20 best ones. The evolutionary algorithm stops when a predefined budget of evaluations is depleted. All simulations are performed using ARGoS3 and the argos3-epuck library under the same conditions and with the same noise levels as in Chocolate—see Section 4.1.1.

4.1.3 EvoComX

EvoComX is an automatic design method derived from EvoCom. The only difference between EvoComX and EvoCom is that EvoComX disregards the $V_b$ vector. The network has only 26 input nodes, instead of the 30 of EvoCom. The four missing nodes are the scalar projection of $V_b$ on the four units vectors.

In the following of the paper, a letter X in the name of a method indicates that the information on the direction from which messages arrive is disregarded—the letter X is the mnemonic for excluded. In our analysis, we include EvoComX—and the X-versions of all methods—to appraise the impact on performance of the information provided by $V_b$. We wish to see whether it is more the advantage of reducing the search space—in this case, number of input nodes— or the disadvantage of discarding the directional information provided by $V_b$.

4.1.4 Gianduja

As already mentioned, Gianduja is the main method that we propose in this paper. It addresses the limitation of Chocolate regarding local communication: in Chocolate, robots cannot explicitly communicate and are only capable of detecting the presence of peers in their neighborhood. In Gianduja, robots are able to locally broadcast one message (one bit of information) and react to it. Messages are sent via the range-and-bearing module by setting a bit of the ping’s payload. Robots can thus identify their neighbors and estimate their position.

Gianduja generates control software for the e-puck platform, as formally specified by RM2, which is the same reference model adopted in EvoCom. Gianduja operates on eight low-level behaviors. Six are the same of Chocolate, extended with a binary parameter $m$: if $m = 1$, the message is broadcast while the behavior is performed; otherwise, it is not. The other two are:

attraction-to-message: the robot goes towards messaging peers;
repulsion-from-message: the robot goes in the opposite direction.

The direction of the messaging peers is provided by $V_b$, as defined in RM2. Behavior transitions can be triggered by eight conditions. Six are the same of Chocolate, the other two are:

- message-count: change state if sufficiently many peers are messaging;
- inverted-message-count: change if they are sufficiently few.

Under the message-count condition, a state transition happens with probability $z(b) = \frac{1}{1+e^{\eta(\xi-b)}}$; under inverted-message-count, the probability is $\bar{z}(b) = 1 - z(b)$. Here, $\eta$ and $\xi$ are parameters of the behavior to be assigned by the design process.

In all other respects, Gianduja is identical to Chocolate. In particular, the two methods adopt the same optimization algorithm with the same parameters, the same simulator with the same noise levels, and the same constraints on how probabilistic finite state machines are assembled.

4.1.5 GiandujaX

GiandujaX is derived from Gianduja. The only difference is that GiandujaX disregards the information provided by $V_b$. This implies that GiandujaX does not use two behaviors of Gianduja: attraction-to-message and repulsion-from-message. However, the robots may still use the conditions message-count and inverted-message-count. In all other respects, GiandujaX is identical to Gianduja.

4.1.6 GiandujaE

GiandujaE is derived from Gianduja and differs from it only in the way received messages are handled. GiandujaE does not use conditions message-count and inverted-message-count. The mechanism for handling the number of messages received is embedded in all remaining conditions, which are extended with two parameters: message ID $\nu$ and threshold $\tau$. Message ID can take two values: if $\nu = 0$, the condition behaves as in Gianduja; if $\nu = 1$, the condition is enabled and behaves as in Gianduja only if the number of neighboring peers that are messaging is larger than the threshold $\tau$, which can take an integer value between 0 and 20.

In the following of the paper, a letter E in the name of the method indicates that received messages are handled as described above—the letter E is the mnemonic for embedded. In principle, this way of handling messages allows more complex behaviors to be obtained. It should be expected that the conjunction of a condition along with the presence of a message allows probabilistic finite state machine of simpler structure, without sacrificing capabilities.
4.1.7 GiandujaEX

GiandujaEX presents the characteristics of GiandujaX and GiandujaE; robots do not use directional information provided by \( V_b \), as in GiandujaX; and implement the transition conditions of GiandujaE. In all other respects, GiandujaEX is identical to Gianduja.

4.2 Experimental Setting

We test and compare the seven design methods described in Section 4.1 on three different missions.

4.2.1 Missions

In all missions, robots operate in a dodecagonal arena of 4.91 m\(^2\). The arena is surrounded by walls. The floor is gray, although some areas may be black or white, depending on the specific mission. Details are provided in the following. All missions are intended to be performed by a swarm of \( N = 20 \) robots within \( T = 120 \) s.

The three missions are aggregation, stop, and decision. We select them because, a priori, one could imagine that communication would play a different role in each of them. Indeed, it is reasonable to expect that aggregation can be solved without the use of communication but that communication can contribute to increasing performance. On the other hand, it is reasonable to expect that decision and stop strictly require communication to be performed. It is also reasonable to expect that the semantics that the optimization would associate to the message is different for each mission.

**AGGREGATION.** Two circular spots of diameter 0.6 m mark the floor of the arena: one is white and the other black. They are positioned on the left-hand half of the arena, 0.25 m apart. At the beginning of each run, the robots are randomly positioned in the right-hand half of the arena so that no robot is on the spots—see Fig. 1(*left*). The robots must aggregate on the white spot as quickly as possible. The black spot is included in the arena to acts as a possible disturbance to the automatic design process. Performance is measured by the following objective function—the higher, the better:

\[
C_A = 24000 - \sum_{t=1}^{T} \sum_{i=1}^{N} I_i(t); \quad I_i(t) = \begin{cases} 
0, & \text{if robot } i \text{ is on the white spot;} \\
1, & \text{otherwise.}
\end{cases}
\]

Performance is non-negative and its theoretical maximum is 24000.
Fig. 1: Arenas for the three missions: AGGREGATION, STOP, and DECISION (from left to right); simulation (top) and real setup (bottom). In DECISION the central spot may be either black or white with equal probability.

**STOP.** One circular white spot with diameter 0.2 m is located near the walls of the arena, in the top-left quadrant. At the beginning of each run, the robots are randomly positioned in the right-hand half of the arena so that none of them is on the spot—see Fig. 1(center). The robots must find the spot as quickly as possible and stop right after. Performance is measured by the following objective function—the higher the better:

$$C_S = 48000 - \left( \bar{t}N + \sum_{t=1}^{\bar{t}} \sum_{i=1}^{N} \bar{I}_i(t) + \sum_{t=\bar{t}+1}^{T} \sum_{i=1}^{N} I_i(t) \right) ;$$

$$I_i(t) = \begin{cases} 1, & \text{if robot } i \text{ is moving;} \\ 0, & \text{otherwise;} \end{cases} \quad \bar{I}_i(t) = 1 - I_i(t).$$

Here, $\bar{t}$ is the time at which the first robot finds the white spot. In the definition of $I_i$ (and $\bar{I}_i$), a robot is moving if it traveled more than 5 mm in the last 100 ms. We adopted this definition to avoid penalizing EvoCom and EvoComX, which are unable to generate behaviors in which robots can stop still, due to the way the outputs of the neural network is encoded. Performance is non-negative and its theoretical maximum is 48000.
**Decision.** One circular spot with diameter 0.6 m is located in the center of the arena. In each experimental run, the spot can be either black or white, with equal probability. A light source is positioned outside the arena, at its right. At the beginning of each run, the robots are randomly positioned in the arena—see Fig. 1(right). The robots must gather in the right-hand half of the arena if the spot is white, or in the left-hand side if it is black. The performance measure—the higher, the better—is:

\[ C_0 = 24000 - \sum_{t=1}^{T} \sum_{i=1}^{N} I_i(t); \]

\[ I_i(t) = \begin{cases} 
0, & \text{if robot } i \text{ is in the correct half of the arena;} \\
1, & \text{otherwise.}
\end{cases} \]

Performance is non-negative and its theoretical maximum is 24000.

### 4.2.2 Protocol

We compare the seven design methods described in Section 4.1 on the three missions described in Section 4.2.1. All experiments involve a swarm of 20 e-puck robots. For each mission, each method is executed 14 times so as to obtain 14 instances of control software. Each execution is allowed a budget of 200,000 simulation runs. We evaluate each method by running the 14 instances of control software it produced once in simulation and once in pseudo-reality—explanation follows. For Chocolate, Gianduja, and EvoCom, the control software produced is run once also on real robots. The order of the experimental runs is randomized to avoid any bias. A pseudo-reality (Ligot and Birattari 2018) is a simulation model, different from the one on which the design process has been performed. It has been shown that experimental runs in pseudo-reality provide indications on the intrinsic ability of a method to cross the reality gap successfully (Ligot and Birattari 2018). The simulation model used as a pseudo-reality for the evaluations of the control software produced is available as supplementary material (Hasselmann and Birattari 2019).

**Statistics.** For each mission, we use boxplots for reporting the performance recorded in simulation using the same model adopted in the design process, in pseudo-reality, and on real robots—when relevant. Simulation results (either on the design model or in pseudo-reality) are represented by gray boxes: thin boxes for the results obtained on the design model and thick ones for those obtained in pseudo-reality. Real-robot results are represented by thick white boxes. Statements on the relative performance of two methods, are supported using a Wilcoxon rank-sum test, at 95% confidence (Conover 1999): any statement like “A performs **significantly** better/worse than B” implies that a Wilcoxon rank-sum test was employed and detected significance with confidence of at least 95%.
We eventually perform a Friedman test (Conover 1999), which aggregates all results by ranking the performance of all methods across all missions. We present the results in a plot that represents the average rank of each method and its 95% confidence interval. The performance of two methods is significantly different if the corresponding intervals do not overlap. In the context of an aggregate analysis, any statements like “A performs significantly better/worse than B” will be based on the observation that the corresponding intervals do not overlap and will be therefore valid with confidence of at least 95%.

In the whole paper, the adjective significant and the adverb significantly are used only to refer to statistical significance. Whenever we use these terms, we imply that an appropriate statistical test (either Wilcoxon or Friedman) has been performed.

4.3 Results

Numerical results, videos, code, and probabilistic finite state machines generated by Chocolate and Gianduja{ø,E,X,EX} are available as supplementary material (Hasselmann and Birattari 2019). In the following, with the notation Gianduja{ø,E,X,EX}, we will collectively refer to Gianduja, GiandujaE, GiandujaX, and GiandujaEX. Similarly, with EvoCom{ø,X} we will refer to EvoCom and EvoComX.

**AGGREGATION.** Results are reported in Fig. 2 (top). Both Gianduja and Chocolate perform significantly better than EvoCom. Although Gianduja is significantly better than Chocolate on the simulation model used for the design, the two perform similarly in reality. Moreover, in pseudo-reality Gianduja{ø,E} are significantly better than Gianduja{X,EX}, whose performance in turn is close to the one of Chocolate. This indicates that Gianduja{ø,E} profitably use $V_b$ to improve performance. Further, the performance of EvoCom and EvoComX is similar, which indicates than EvoCom fails to benefit from the directional information provided by $V_b$. In this particular mission, there is no evidence that GiandujaE improves over Gianduja. EvoCom{ø,X} suffer more from the (pseudo-)reality gap than the other methods. Gianduja suffers more from the reality gap than Chocolate: this could be due to the uncertainty of $V_b$; although the ability to detect where a message originates is a potential advantage, it becomes a burden due to uncertainty.

At visual inspection, in Chocolate and Gianduja{EX,X} robots navigate randomly in the arena and stop whenever they enter the white spot. In Gianduja{ø,E}, when this happens, robots start broadcasting the message to attract neighboring robots. In EvoCom{ø,X} robots randomly explore the arena until they enter the white spot. When this happens, they spin in place. Videos are available in Hasselmann and Birattari (2019).
Fig. 2: AGGREGATION, STOP, and DECISION (from top to bottom). Thick white boxes represent the results of robot experiments; thick gray boxes those of pseudo-reality; thin gray ones, those of simulations performed on the basis of the same simulation model adopted in the design process. The higher the better.
Fig. 3: Example of probabilistic finite state machine produced by Gianduja for the three aforementioned missions: aggregation (A), stop (B), and decision (C).
In this mission, pseudo-reality gives a reasonably good indication on the performance in reality of EvoCom and Chocolate but overestimates the performance of Gianduja.

An example of probabilistic finite state machine produced by Gianduja is reported in Fig. 3(A). The robots start in the exploration state. If they step on white ground, they can transition to stop and start broadcasting the message. Robots that are still in the exploration state and receive a sufficiently large number of messages to trigger the message-count condition transition to the attraction to message behavior. As a result, they will approach the robots that broadcast the message, eventually reaching the white spot. Seen from the point of view of the sender, the semantics given to the message is: I am on the white spot; while, seen from the point of view of the receiving robots, it is: Go there.

**STOP.** Results are reported in Fig. 2(*middle*). Gianduja performs significantly better than EvoCom and Chocolate on the simulation model used for the design and in reality. The performances of Gianduja{E,EX,X} are similar both on the simulation model used for the design and in pseudo-reality. In this mission, there is no evidence that any of them improves over Gianduja. Indeed, Gianduja performs significantly better than GiandujaE and GiandujaX in pseudo-reality.

At visual inspection, EvoCom{ø,X} appear to be unable to use communication effectively whereas Gianduja{ø,E,X,EX} do. In EvoCom{ø,X} robots move randomly until their motion is stopped by the walls of the arena. This behavior is qualitatively similar to the one of Chocolate, which cannot leverage communication. In Gianduja{ø,E,X,EX}, robots move randomly until one of them enters the white spot, when this happens the robot stops and starts sending the message. Other robots then stop and relay the message. Videos are available in Hasselmann and Birattari (2019).

Chocolate and EvoCom suffer the reality gap more than Gianduja. The performance in pseudo-reality gives a good indication on the relative drop in performance of the three methods but, in this case, fails to predict the high variance that we observe in reality.

An example of probabilistic finite state machine produced by Gianduja is reported in Fig. 3(B): the robots start by performing exploration. They transition to stop either if they step on white floor or if they receive the message from their peers. While in stop, they broadcast the message. Seen from the point of view of the sender, the semantics given to the message is: I found the white spot; while, seen from the point of view of the receiving robots, it is: Stop.

**DECISION.** Results are reported in Fig. 2(*bottom*). Gianduja performs significantly better than EvoCom{ø,X} and Chocolate on the simulation model used for the design, in pseudo-reality, and in reality. In pseudo-reality, GiandujaE is slightly but significantly better than Gianduja{ø,EX}. Also, the inter-quartile range observed in this experiment is smaller for GiandujaE than for the other
two methods and for GiandujaX. There is no clear evidence that using information provided by the vector $V_b$ helps, nor that the increased search space hinders performance. Both GiandujaE and Gianduja are significantly better than Chocolate in pseudo-reality. This indicates that, in this mission, communication contributes to performances and Gianduja[$g,E$] leverage it effectively.

At visual inspection, it appears that Gianduja[$g,E,X,EX$] use communication meaningfully. In all these methods, every realization of the design process selects a default behavior: by default, at the beginning of a run, the robots either go towards the light or away from it. While doing so, some robots by chance enter the central spot: if the color does not correspond to the default behavior, they start broadcasting the message and revert their direction of motion. Robots that receive the message relay it and revert their direction of motion as well. In Chocolate, to make an informed decision, each individual robot needs to enter the central spot. In EvoCom[$g,X$], robots appear to randomly select one side or the other of the arena. As a result, performance is very inconsistent. Videos are available in Hasselmann and Birattari (2019).

Gianduja[$g,E,X,EX$] cross the (pseudo-)reality gap in a satisfactory way. In this mission, pseudo-reality gives a good indication of the performance in reality.

An example of probabilistic finite state machine produced by Gianduja is reported in Fig. 3(C): the robots start in the anti-phototaxis behavior. They transition to phototaxis either if they step on white floor or if they receive the message from their peers. While in phototaxis, they broadcast the message. Seen from the point of view of the sender, the semantics given to the message is: _I stepped on the white spot_; while, seen from the point of view of the receiving robots, it is: _Go towards the light_.

### 4.3.1 Aggregate Results

The aggregate results obtained in real-robot experiments are presented in Fig. 4; those obtained in pseudo-reality are presented in Fig. 5. Across all
missions, the Friedman test indicates that, on the robots, the control software generated by Gianduja performs significantly better than the one of Chocolate and EvoCom. In pseudo-reality, Gianduja performs significantly better that Gianduja\{EX,X\}, EvoCom\{\emptyset,X\}, and Chocolate. All in all, on the aggregate data, pseudo-reality provides a correct prediction of the relative performance in robot experiments.

The aggregate results over the three missions considered confirm that communication improves performance and that Gianduja leverages it effectively yielding significantly better results than EvoCom. At least in the setting considered in our experiments, EvoCom does not appear to be able to evolve effective behaviors. The performance of the behavior produced by EvoCom is worse than the one produced by Chocolate, which by construction do not use communication. This can be considered as a major failure. The results on the three missions considered fail to provide any evidence that Gianduja\{E\} is better than Gianduja. On the other hand, they show that Gianduja performs significantly better than Gianduja\{X,EX\}.

5 Study B: Communication with two or three bits of information.

In this section, we study the automatic design of communication when robots can broadcast and react to multiple messages. We introduce a number of automatic design methods that extend those we described in study A by enabling the possibility of exchanging more information. Specifically, we introduce Gianduja\{2,2E\} and Gianduja\{3,3E\}, which extend Gianduja\{\emptyset,E\} with the ability to broadcast and react to two bits or three bits of information, respectively. Gianduja\{2,2E\} are based on reference model RM 2.1-2; Gianduja\{3,3E\}, on RM 2.1-3—see Table 3.

We evaluate these methods on new missions that we think require the use of at least two different messages. The goal of the experiments is to assess whether Gianduja, and more generally the ideas presented in the paper, can possibly scale up to more complex missions that require more information to
Table 3: Reference model RM 2.1-ℓ. Novelties with respect to RM 2 are highlighted. RM 2.1-ℓ is a parametric family of models that differs one from the other in the amount of information that robots can exchange. The parameter ℓ identifies a specific model within the family by defining the number of bits of information that robots can exchange: in RM 2.1-2, robots can exchange two bits of information; in RM 2.1-3, three. RM 2.1-ℓ is a proper extension of RM 2: for ℓ = 1, RM 2.1-1 ≡ RM 2.

<table>
<thead>
<tr>
<th>Input</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>proxᵢ ∈ {1,...,8}</td>
<td>[0,1]</td>
<td>reading of proximity sensor i</td>
</tr>
<tr>
<td>lightᵢ ∈ {1,...,8}</td>
<td>[0,1]</td>
<td>reading of light sensor i</td>
</tr>
<tr>
<td>gndⱼ ∈ {black, gray, white}</td>
<td></td>
<td>reading of ground sensor j</td>
</tr>
<tr>
<td>n</td>
<td>[0,20]</td>
<td>number of neighboring robots perceived</td>
</tr>
<tr>
<td>V</td>
<td>([0.5, 20], [0, 2π])</td>
<td>their relative aggregate position</td>
</tr>
<tr>
<td>bⱼ ∈ {0,2ℓ−1}</td>
<td>[0,20]</td>
<td>number of neighbors broadcasting message h</td>
</tr>
<tr>
<td>Vⱼ ∈ {0,2ℓ−1}</td>
<td>([0.5, 20], [0, 2π])</td>
<td>their relative aggregate position</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Output</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vₖ ∈ {l,r}</td>
<td>[−0.12, 0.12] m/s</td>
<td>target linear wheel velocity</td>
</tr>
<tr>
<td>sⱼ ∈ {on, off}</td>
<td></td>
<td>broadcast state for message h</td>
</tr>
</tbody>
</table>

Period of the control cycle: 100 ms

be exchanged by the robots. We also wish to see if the way in which messages are handled by GiandujaE increases performance with respect to Gianduja when missions are more complex than those considered in study A. Finally, we wish to understand whether the design methods proposed in the paper are able to handle the larger search space that results from increasing the number of messages that robots can broadcast and to which they can react.

5.1 Eight design methods

In this section, we introduce eight new design methods based on Gianduja, GiandujaE, and EvoCom. Here, we drop Gianduja{X,EX} because in study A they performed significantly worse than Gianduja.

5.1.1 Gianduja2

Gianduja2 extends Gianduja by allowing the communication of two bits of information, which amounts to the ability to broadcast and react to $2^2 - 1 = 3$ different messages. The term $-1$ accounts for the possibility of not sending any message. Gianduja2 operates on the same low-level behaviors of Gianduja, with only a minor modification. Here, the parameter $m$ can take four integer values: $\{0,1,2,3\}$. If $m = 0$, the robot does not broadcast any message—or, more precisely, it broadcasts via its range-and-bearing module the null message that only allows its neighboring peers to detect its presence, without conveying any extra signification. If $m = \{1,2,3\}$, the robot broadcasts...
Table 4: Binary encoding of messages adopted in EvoCom2—white background: 2 bits for the 4 messages, including the null one. Encoding adopted in EvoCom3—white and gray background: 3 bits for the 8 messages, including the null one.

<table>
<thead>
<tr>
<th>h</th>
<th>bit2(h)</th>
<th>bit1(h)</th>
<th>bit0(h)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

the corresponding message, to convey the meaning that the optimization process will have associated with the message itself, for the specific mission at hand. Moreover, in Gianduja2, the two behaviors attraction-to-message and repulsion-from-message take one additional parameter $m \in \{1, 2, 3\}$, which specifies the message to whose senders the robot is attracted, or from whose senders it is repelled. Finally, the two conditions message-count and inverted-message-count take one additional parameter $m \in \{1, 2, 3\}$, which specifies the message to which the condition should be sensitive. The transition probability is computed via $z(\cdot)$, as defined in Gianduja.

5.1.2 Gianduja2E

Gianduja2E derives from Gianduja2, in the same sense in which GiandujaE derives from Gianduja. Like GiandujaE, Gianduja2E does not use the conditions message-count, and inverted-message-count. Also in Gianduja2E, the conditions black-floor, white-floor, gray-floor, neighbor-count, inverted-neighbor-count, and fixed-probability are extended with two parameters: message ID $\nu$ and threshold $\tau$. Parameter $\nu$ can take four different values: if $\nu = 0$, the condition behaves as in Gianduja2; if $\nu = \{1, 2, 3\}$, the condition is enabled and behaves as in Gianduja2 if the number of neighbors sending message 1, 2, 3, respectively, is greater than $\tau$.

5.1.3 EvoCom2

EvoCom2 extends EvoCom by allowing the communication of two bits of information. With respect to EvoCom, EvoCom2 has five additional inputs and one additional output: the network totalizes 35 inputs and 4 output nodes. The five additional inputs are used to encode information concerning the messages received. In EvoCom, five inputs are devoted to this; in EvoCom2, they are ten. Two are computed on the basis of the number of neighboring peers that broadcast the various messages available, and the remaining eight on the direction from which messages arrive. The additional output is used to encode
the broadcast state. In EvoCom, one output is devoted to this; in EvoCom2, they are two.

Before we can formally describe how these inputs and outputs are computed, we need to define some notation. In EvoCom2, messages are identified via a binary representation. Table 4 provides a mapping between a message ID \( h \) and its binary representation. Let us define the function \( \text{bit}_p(h) \) that returns TRUE/1, if the \( p \)-th bit of the binary representation of \( h \) is set; and FALSE/0, otherwise. In EvoCom2, \( p \in \{0, \ldots, \ell-1\} \), with \( \ell = 2 \). Bit \( \text{bit}_p = 0 \) is the less significant one. For \( p \in \{0, \ldots, \ell-1\} \): let \( \gamma_p = \sum_{h: \text{bit}_p(h) = 0} b_h \) be the number of neighboring peers broadcasting any message \( h \) whose binary representation has the \( p \)-th bit set; and let \( \Gamma_p = \sum_{h: \text{bit}_p(h) = 1} V_h \), be the composition of vectors \( V_h \) concerning any message \( h \) whose binary representation has the \( p \)-th bit set.

With this notation, we can represent the ten inputs devoted to encoding information concerning the messages received: two are \( z(\gamma_p) = 1 - \frac{2}{1 + e^{\gamma_p}} \), with \( p \in \{0, \ldots, \ell-1\} \); eight are the projections of \( \Gamma_p \), with \( p \in \{0, \ldots, \ell-1\} \), onto the unit vectors pointing at 45°, 135°, 225°, and 315° with respect to the front of the robot. The two outputs determine which message should be broadcast according to the binary representation given in Table 4.

5.1.4 EvoCom2X

EvoCom2X is an automatic design method derived from EvoCom2, the only difference between the two is that EvoCom2X disregards the \( \Gamma_p \) vector. The network has only 27 input nodes, instead of the 35 of EvoCom2. The eight missing input node are the projections of \( \Gamma_p \).

5.1.5 Gianduja3

Gianduja3 extends Gianduja2, and eventually Gianduja, by allowing the communication of three bits of information, which amounts to the ability to broadcast and react to \( 2^3 - 1 = 7 \) different messages. The term −1 accounts for the possibility of not sending any message. Gianduja3 is defined as Gianduja2, with the difference that the parameter \( m \) can take eight integer values, \( \{0, \ldots, 7\} \); and the parameter \( \mathbb{m} \) can take seven: \( \{1, \ldots, 7\} \).

5.1.6 Gianduja3E

Gianduja3E derives from Gianduja3, in the same sense in which GiandujaE and Gianduja2E derive from Gianduja and Gianduja2, respectively. Gianduja3E is defined as Gianduja2E, with the difference that the parameter \( \nu \) can take eight integer values, \( \{0, \ldots, 7\} \).

5.1.7 EvoCom3

EvoCom3 extends EvoCom2, and eventually EvoCom, by allowing the communication of three bits of information. It is defined as EvoCom2, with \( \ell = 3 \).
Here, the total number of inputs is 40 and the total number of output is 5. The number of inputs used to encode the number of neighboring peers that broadcast the various messages available is $\ell = 3$. The number of inputs used to encode the direction from which messages arrive is $4\ell = 12$. The number of outputs used to encode the broadcast state is $\ell = 3$.

5.1.8 EvoCom3X

EvoCom3X is an automatic design method derived from EvoCom3, the only difference between the two is that EvoCom3X disregards the $\Gamma_p$ vector. The network has only 28 input nodes, instead of the 40 of EvoCom3. The twelve missing input node are the projections of $\Gamma_p$.

5.2 Experimental Setting

We test and compare Gianduja\{2,2E,3,3E\} and EvoCom\{2,2X,3,3X\} on 3 different missions.

5.2.1 Missions

In all three missions, $N = 20$ robots operate in the same dodecagonal arena described in Section 4.2.1. As in study A, the amount of time available to the robots for performing each mission is $T = 120s$. The three missions are beacon aggregation, beacon stop, and beacon decision. They were selected because we thought that, to be successfully performed, they require the use of multiple messages with different semantics. In all three missions, there is a beacon that can broadcast a message to which robots must react. The role of the beacon is played by an e-puck robot that does not move throughout the experiment. It is not considered to be part of the swarm.
**BEACON AGGREGATION.** Two circular spots of diameter 0.6 m mark the floor of the arena: one is white and the other black. They are positioned on the left-hand half of the arena, separated by 0.25 m—see Fig 6(left). A beacon positioned between the two spots broadcasts either message 1 or 2 for the whole duration of a run. The message to be broadcast is selected randomly with equal probability at the beginning of each run. At the beginning of each run the robots are randomly positioned in the right-hand half of the arena so that no robot is on either spot. Robots must aggregate on the black spot, if the beacon broadcasts 1; on the white one, if it broadcasts 2.

Performance is measured by the following objective function—the higher the better:

\[
C_{BA} = 24000 - \sum_{t=1}^{T} \sum_{i=1}^{N} I_i(t);
\]

\[
I_i(t) = \begin{cases} 0, & \text{if robot } i \text{ is on the correct spot;} \\ 1, & \text{otherwise.} \end{cases}
\]

Performance is non-negative and its theoretical maximum is 24000.

**BEACON STOP.** The arena’s floor is gray and a beacon is positioned in the middle of the arena—see Fig 6(center). The beacon broadcasts message 1 or 2 after a time uniformly sampled in [40 s; 60 s]—the message to be broadcast is selected randomly with equal probability at every run. At the beginning of each run, the robots are randomly positioned in the arena. Robots must stop as soon as the beacon starts broadcasting either one of the two messages.

Performance is measured by the following objective function—the higher the better:

\[
C_{BS} = 24000 - \left( \sum_{t=1}^{\bar{t}} \sum_{i=1}^{N} \bar{I}_i(t) + \sum_{t=\bar{t}+1}^{T} \sum_{i=1}^{N} I_i(t) \right);
\]

\[
I_i(t) = \begin{cases} 1, & \text{if robot } i \text{ is moving;} \\ 0, & \text{otherwise;} \end{cases}
\]

\[
\bar{I}_i(t) = 1 - I_i(t).
\]

In this equation, \( \bar{t} \) is the time at which the beacon starts broadcasting a message. In the definition of \( I_i \) (and \( \bar{I}_i \)), a robot is considered moving if it traveled more than 5 mm in the last 100 ms. This accounts for the inability of EvoCom\{2,2X,3,3X\} to generate behaviors in which robots can stop still. Performance is non-negative and its theoretical maximum is 24000.

**BEACON DECISION.** The floor of the right-hand quarter of the arena is black; the rest is gray. A light source is positioned outside the arena on its right-hand side—see Fig 6(right). A beacon is positioned at the middle of the interface between the black and the gray parts and acts as a timed trigger. At the beginning of a run, the beacon starts broadcasting either message 1 or 2—randomly selected with equal probability. After 60 s, the beacon switches to
the other message. At the beginning of each run, the robots are randomly positioned in the arena.

Robots must position themselves either on the black or on the gray part of the arena, depending on the message broadcast by the beacon: on the black, if the message is 1; on the gray, if it is 2. The performance of the robots is measured by the following objective function—the higher, the better:

$$C_{\text{bin}} = 24000 - \sum_{t=1}^{T} \sum_{i=1}^{N} I_i(t);$$

where

$$I_i(t) = \begin{cases} 0, & \text{if robot } i \text{ is in the correct side of the arena;} \\ 1, & \text{otherwise.} \end{cases}$$

Performance is non-negative and its theoretical maximum is 24,000.

### 5.2.2 Protocol

The protocol is similar to the one of study A, as described in Section 4.2.2. We compare the eight design methods described in Section 5.1 on the three missions described in Section 5.2.1. All experiments involve a swarm of 20 e-puck robots. For each mission, each method is executed 14 times so as to obtain 14 instances of control software. Each execution is allowed a budget of 200,000 simulation runs. We evaluate each method by running the 14 instances of control software it produced once in simulation and once in pseudo-reality—see Section 4.2.2 for an explanation.

**Statistics.** Results are analyzed and reported as described in Section 4.2.2.

### 5.3 Results

We present the results on a per-mission basis and then we aggregate them across the three missions. Numerical results, videos, code, and probabilistic finite state machines generated by Gianduja{2,2E,3,3E} are available as supplementary material (Hasselmann and Birattari 2019).

**Beacon Aggregation.** Results are reported in Fig. 7(top). Gianduja and all its variants perform significantly better than EvoCom{2,2X,3,3X} in pseudo-reality. All methods derived from Gianduja also appear to be less affected by the pseudo-reality gap than EvoCom{2,2X,3,3X}. At visual inspection, all methods derived from Gianduja perform similarly both in simulation and pseudo-reality. Although the robots aggregate on the spots, they do not comply to the instructions provided by the beacon and split evenly between the two spots. Videos are available in Hasselmann and Birattari (2019). The poor performance on this missions may be due to the complexity of the task and the absence of any memory component in all the control architectures considered.
Fig. 7: beacon aggregation, beacon stop, and beacon decision (from top to bottom). Thick gray boxes represent the results in pseudo-reality; thin gray ones, those of simulations performed on the same simulation model adopted in the design process. The higher, the better.
Fig. 8: Example of probabilistic finite state machine produced by Gianduja2E for the three missions: BEACON AGGREGATION (A), BEACON STOP (B), and BEACON DECISION (C).
An example of probabilistic finite state machine produced by Gianduja2E is reported in Fig. 8(A): as the robots start on gray floor, they immediately transition from stop to exploration. They then transition back to stop whenever they find either the white or the black spot, no matter the message broadcast by the beacon. In this mission, the semantics assigned to the message by the optimization algorithm is unclear. The robots fail to use their messaging capabilities effectively. Our conjecture is that the probabilistic finite state machines adopted by Gianduja{2,2E,3,3E} are not sufficiently expressive: the maximum number of states and transitions allowed in the design process is too small. We also conjecture that the duration of the experiment is too short for the robots to be able to accomplish the mission. We elaborate on these conjectures in appendix A.

**Beacon stop.** Results are reported in Fig. 7(middle). Gianduja{2E,3E} outperform all other methods both in simulation and pseudo-reality. Gianduja{2,3} perform significantly better than EvoCom{2,2X,3,3X} in pseudo-reality. EvoCom{2,2X,3,3X} suffer greatly from the pseudo-reality gap, Gianduja{2,3} are less affected, and Gianduja{2E,3E} are almost not affected at all. At visual inspection, all methods derived from Gianduja appear to correctly associate the messages broadcast by the beacon to stop. The structure of the state machines produced by Gianduja{2E,3E} are simpler, more robust, and produce better results than those produced by Gianduja{2,3}. In Gianduja{2,2E,3,3E}, the robots randomly explore the arena until they receive a message from the beacon. When this happen, they stop and start relaying the message. All in all, the robots appear to use their communication capabilities effectively. Videos are available in Hasselmann and Birattari (2019). As in stop of study A, EvoCom{2,2X,3,3X} produce behaviors in which robots move randomly until their motion is stopped by the walls of the arena. Although trivial, this behavior performs reasonably well because, due to the size of the arena, the typical time needed by the robots to reach the walls is close to \(t\). Based on this observation, we conjecture that the performance of the behaviors generated by EvoCom{2,2X,3,3X} might not be robust to variations in the arena size. We elaborate on this conjecture in appendix A, where we compare the behaviors generated by EvoCom2 and Gianduja2E in arenas of different sizes.

For all methods considered, the performance of the three-bit versions is relatively close to the one of their two-bit counterparts. We can therefore conclude that, in this mission, the respective optimization algorithms can effectively search the larger solution space that results from extending the amount of information exchanged to three bits.

An example of probabilistic finite state machine produced by Gianduja2E is reported in Fig. 3(B): the robots start in phototaxis—which, in the absence of a light source, reduces to exploration. They transition to stop whenever they receive message 1 from the beacon and the floor is gray (which is always true) or when they receive message 2 and sufficiently few robots are in the neighborhood (which is easily achieved considering the parameters of the behavior).
Table 5: Relative median performances of Gianduja\{3,3E\} and EvoCom2 with respect to Gianduja2 and Gianduja2E.

<table>
<thead>
<tr>
<th></th>
<th>Gianduja3</th>
<th>EvoCom2</th>
<th>Gianduja3E</th>
<th>EvoCom2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Beacon aggregation</td>
<td>−11.7%</td>
<td>−83.7%</td>
<td>−18.9%</td>
<td>−83.0%</td>
</tr>
<tr>
<td>Beacon stop</td>
<td>−12.9%</td>
<td>−25.7%</td>
<td>−1.06%</td>
<td>−35.5%</td>
</tr>
<tr>
<td>Beacon decision</td>
<td>+7.44%</td>
<td>−2.89%</td>
<td>−3.10%</td>
<td>−5.23%</td>
</tr>
</tbody>
</table>

While in the stop state, robots broadcast message 2 to trigger the transition on other robots. Seen from the point of view of the sender, the semantics given to the message is: *I received a message (I am relaying it)*; while, seen from the point of view of the receiving robots, it is: *Stop*.

**Beacon decision.** Results are reported in Fig. 7(bottom). EvoCom\{2,3\} suffer greatly from the pseudo-reality gap.

At visual inspection, Gianduja\{2E,3E\} and EvoCom\{2,2X,3,3X\} appear to make an effective use of communication. Depending on the message broadcast by the beacon, the robots are either attracted or repulsed by the light and thus go towards or away from the black area. At time $T = t$, when the beacon switches message, the robots invert their behavior. Videos are available in Hasselmann and Birattari (2019).

Contrary to Gianduja\{2E,3E\}, Gianduja\{2,3\} appear to be unable to generate swarms that can switch their behavior and leave the black area when requested to do so by the beacon. At least in this mission, Gianduja\{2,3\} are unable to produce behaviors as rich and expressive as those produced by Gianduja\{2E,3E\}. Also in this mission, the increase of the number of bits is not detrimental to the performance: the performance of Gianduja3 (Gianduja3E) is close to the one of Gianduja2 (Gianduja2E).

An example of probabilistic finite state machine produced by Gianduja2E is reported in Fig. 3(C): the robots start in phototaxis. If the beacon broadcasts message 1, the robots directly transition to anti-phototaxis, relay message 1, and oscillate back and forth—they do so by cycling through phototaxis, anti-phototaxis, and exploration. Otherwise, if the beacon broadcasts message 2, the robots cycle through phototaxis and exploration, depending on the number of neighbors and the color of the floor. By doing so they leave the black area. In this case, the state machine produced by Gianduja2E is hardly readable due to the high number of transitions. It is however possible to infer a semantics for messages 1 and 2: Seen from the point of view of the sender, the semantics is: *I received message 1 (or 2)*; while, seen from the point of view of the receiving robots, it is: *Go away from the light for message 1 and Go towards the light for message 2.*
5.3.1 Aggregate Results

Figure 9 reports the aggregate results in pseudo-reality. They confirm that Gianduja\{2,2E,3,3E\} perform significantly better than EvoCom\{2,3,2X,3X\}. The results also show that the addition of a third bit of data is not needed to perform the three missions considered: indeed, Gianduja\{2,2E\} perform better than Gianduja\{3,3E\}. Nonetheless it is interesting to observe that Gianduja could handle nicely the increased search space resulting from the addition of the third bit. Although the performance of Gianduja\{3\} (Gianduja\{3E\}) is lower than the one of Gianduja\{2\} (Gianduja\{2E\}), the drop is relatively small: in Table 5, we report the relative median performance of Gianduja\{3\} with respect to Gianduja\{2\} and we compare it with the one of EvoCom2.

6 Conclusions

In the paper, we studied the automatic design of control software for a swarm of robots capable of local communication. Specifically, we studied the emergence of a semantics for an a priori meaningless message: the automatic design process is expected to assign an appropriate semantics to a message—intended here as a pure signifier—based on the specific mission to be performed. We introduced Gianduja and its variants, a family of automatic design methods that produce control software in the form of a probabilistic finite state machine, by assembling predefined behavioral modules that have been previously defined in a mission-agnostic way. In the experiments presented in the paper, Gianduja and its variants were able to find solutions that revealed, at visual inspection, to properly leverage the communication capabilities of the robots by assigning a meaningful semantics to the messages. With a single exception, this holds true for all the missions we considered. We compared Gianduja and its variants with EvoCom, a neuro-evolutionary approach. In a first study, we considered robots that can exchange one bit of information and we compared them on three missions: Gianduja and its variants outperformed EvoCom in
all missions. We then considered robots that can exchange two and three bits of information and we defined variants of Gianduja, Gianduja\{2,2\} and Gianduja\{3,3\}, that can produce control software for robots endowed with this capability. We tested these variants on three other missions: Gianduja\{2,2\} performed better than EvoCom on two out of three missions. When aggregated, the results we obtained show that Gianduja is the best method for one bit of information and Gianduja\{2,2\} is the best one for missions requiring more bits of information to be exchanged. Our results do not provide any conclusive evidence on which variant of Gianduja is the best one: further investigation is needed to understand the impact of their underlying design choices. Future work will focus on studying the relationship between the complexity of a mission and the number of bits of information required to solve it. Finally, although the results of the study we performed using a pseudo-reality indicate that Gianduja and its variants are relatively robust to the reality gap, further robots experiments should be performed to reliably characterize the performance of these methods.

A Appendix

In this appendix we present additional results to support the conjectures made in study B on why Gianduja fails to produce satisfactory results.

**BEACON AGGREGATION.** To support our conjecture that Gianduja produces poor results because the finite state machines adopted are not sufficiently expressive, we present an improved version of Gianduja\{2,2\} that we shall call Gianduja\{2,2\}. In Gianduja\{2,2\}, the automatic design process is allowed to generate finite state machine comprising up to 6 states and up to 6 outgoing edges from each state. We also increase the duration of an experimental run to 240 s. At visual inspection, the performance of the control software produced by Gianduja\{2,2\} obtains satisfactory results and, in most cases, robots aggregate on the correct spot. Videos are available in Hasselmann and Birattari (2019). Results reported in Fig. 10 show that control software generated by Gianduja\{2,2\} performs similarly in the two scenarios.

**BEACON STOP.** To support our conjecture on scalability, we test the control software generated by EvoCom\{2\} and Gianduja\{2,2\} in arenas of larger surface and with a larger number of robots—scale factors: 2, 4, 8, and 16. Results are reported in Fig. 11. They show that, both in simulations performed with the design model and in pseudo-reality, Gianduja\{2,2\} yields good performance for all considered scaling factors. On the other hand, the performance of EvoCom\{2\} drops slightly in pseudo-reality (where it was already relatively low) and largely on the design model.

**Acknowledgements.**

The authors thanks Frédéric Robert for contributing to the definition of the missions of study A. The project has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No 681872). Mauro Birattari acknowledges support from the Belgian Fonds de la Recherche Scientifique – FNRS.
Fig. 10: beacon aggregation. $\text{Gianduja}2E'$: the black line represents the cumulative distribution of the results in simulation. The red and blue line represent the cumulative distributions restricted to each of the two possible scenarios.

References


Fig. 11: BEACON STOP with different scaling factors. Thick gray boxes represent results in pseudo-reality; thin white ones, those of simulations performed on the basis of the same simulation model adopted in the design process. The higher, the better.
In: Swarm Intelligence, 9th International Conference, ANTS, Springer, Berlin, Germany, vol 8667, pp 25–37
Automatic design of robots swarms with communication capabilities


