PUC

ISSN 0103-9741

Monografias em Ciência da Computação
n° 06/2017

Engineering Smart Things based on Evolved Networks

Nathalia Moraes do Nascimento
Carlos José Pereira de Lucena

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453-900

RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 06/2017
Editor: Prof. Carlos José Pereira de Lucena
ISSN: 0103-9741
Abril, 2017

Engineering Smart Things based on Evolved Networks

Nathalia Moraes do Nascimento and Carlos José Pereira de Lucena

nnascimento@inf.puc-rio.br, lucena@inf.puc-rio.br

Abstract. The goal of the Internet of Things (IoT) is to transform any thing around us, such as a trash can or a street light, into a smart thing. A smart thing has the ability of sensing, processing, communicating and/or actuating. In order to achieve the goal of a smart IoT application, such as minimizing waste transportation costs or reducing energy consumption, the smart things in the application scenario must cooperate with each other without a centralized control. Inspired by known approaches to design swarm of cooperative and autonomous robots, we modeled our smart things based on the embodied cognition concept. Each smart thing is a physical agent with a body composed of a microcontroller, sensors and actuators, and a brain that is represented by an artificial neural network. This type of agent is commonly called an embodied agent. The behavior of these embodied agents is autonomously configured through an evolutionary algorithm that is triggered according to the application performance. To illustrate, we have designed three homogeneous prototypes for Smart Street Lights based on an evolved network. This application has shown that the proposed approach results in a feasible way of modeling decentralized smart things with self-developed and cooperative capabilities.

Keywords: embodied cognition, evolved network, smart things, internet of things, self-developed systems, cooperative systems, emergent communication system

Resumo. A Internet das Coisas (sigla “IoT"de “Internet of Things"em inglês) visa transformar qualquer coisa do nosso dia-a-dia, a exemplo de lixeiras e postes públicos, em coisas inteligentes (“smart things"). Um “smart thing"tem a habilidade de coletar e processar dados do ambiente, comunicar e/ou agir. De forma a alcançar o objetivo geral de uma aplicação em IoT, a exemplo de minimizar os custos com a coleta de lixo ou reduzir o consumo de energia, os “smart things"no cenário de aplicação devem cooperar uns com os outros sem que haja um ponto central para coordenar essas interações. Inspirado em abordagens existentes para modelar grupos de robôs autônomos e cooperativos, os “smart things"foram modelados com base no conceito de cognição incorporada (“embodied cognition"). Cada “smart thing"é um agente físico com um corpo composto por um microcontrolador, sensores e atuadores, e por um “cérebro"que é representado por uma rede neural artificial. Este tipo de agente é comumente chamado de agente incorporado (“embodied agent"). O comportamento desses agentes é automaticamente configurado através de um algoritmo evolutivo, que é ativado de acordo com a performance da aplicação. Para ilustrar o uso do modelo, três protótipos de postes públicos inteligentes foram desenvolvidos fazendo uso de uma rede neural evoluída. Como resultado, essa aplicação mostrou que fazendo uso dessa abordagem , é possível modelar um conjunto decentralizado de “smart things"com habilidades de autodesenvolvimento e de cooperação.

Palavras-chave: cognição incorporada, rede neural evoluída, coisas inteligentes, internet das coisas, sistemas autodesenvolvidos, sistemas cooperativos, sistema de comunicação emergente

Responsável por publicações:
Rosane Teles Lins Castilho
Assessoria de Biblioteca, Documentação e Informação
PUC-Rio Departamento de Informática
Rua Marquês de São Vicente, 225 - Gávea
22453-900 Rio de Janeiro RJ Brasil
Tel. +55 21 3114-1516 Fax: +55 21 3114-1530
E-mail: bib-di@inf.puc-rio.br
Web site: http://bib-di.inf.puc-rio.br/techreports/

Sumário

1 Introduction

A few years ago, Kephart and Chess (2003) [1] called the global goal to connect trillions of computing devices to the Internet the nightmare of ubiquitous computing. The reason for that is that to reach this global goal requires a lot of skilled Information Technology (IT) professionals to create millions of lines of code, and install, configure, tune, and maintain these devices. According to Kephart (2005) [2], in some years, IT environments will be impossible to be administered, even by the most skilled IT professionals.

Predicting the emergence of this problem, in 2001 the IBM company suggested the creation of autonomic computing [3]. IBM recognized that the only viable solution to resolve this problem was to endow systems and the components that comprise them with the ability to manage themselves in accordance with high-level objectives specified by humans [2]. Therefore, IBM proposed systems with self-developed capabilities. The company emphasized the need of automating IT key tasks, such as coding, configuring, and maintaining systems, based on the progress observed in the automation of manual tasks in agriculture.

Other IT companies agreed with IBM and then generated their own manifests, such as Hewlett-Packard [4] and Microsoft [5]. However, the IT Industry interest in the development of self-management devices is not yet very evident. As a result, not only has the goal of the Internet of Things (IoT) to connect billions of devices to the Internet not been reached, but also we have been experiencing the problems previously listed by Kephart and Chess (2003) [1].

The truth is that companies and researchers are so busy competing to define the official protocol and architecture for the Internet of Things, that very little research has been done to provide a sophisticated control to manage all these billions of things. As a result, there is a lack of software to support the development of a huge number of different IoT applications.

In this context, we have been investigating how to create applications based on the Internet of Things with self-developed capabilities. To this end, our approach consists in:

- Developing Smart Things:
  - Things that are autonomous and able to cooperate and execute complex behavior without the need for centralized control to manage their interaction.
  - Things that are able to have behavior assigned at design-time and/or at run-time.
- Providing mechanisms to allow things to self-adapt and to improve their own behavior;

To reach these objectives, we previously developed a generic software basis for IoT, which is called the "Framework for Internet of Things"(FIoT) [6]. The framework approach was used to develop the common requirements among IoT applications and implement a reusable architecture [7]. We developed FIoT according to the following directions:

1. To create autonomous things and a distributed control, we modeled the framework based on a multi-agent approach [8]. According to Cetnarowicz et. al (1996) [8], the active agent was invented as a basic element from which distributed and decentralized systems could be built. In our approach, we considered the use of embodied agents,

which is typically used to model and control autonomous physical objects that are situated in actual environments [9].

2. To control the things, we chose a control architecture based on artificial neural networks [10]. A neural network is a well known approach to dynamically provide responses and automatically create a mapping of input-output relations [10]. In addition, it is commonly used as an internal controller of embodied agents [11, 12].

3. To make things self-adaptive, we proposed the use of the IBM control-loop [13] combined with various Machine Learning (ML) techniques, notably supervised learning and evolutionary algorithms [14].

As the development of smart things is part of a broader context, a set of related aspects will be left out of the scope of this work. Thus, the following approaches are not directly addressed by this work: security, ontology, protocols and scalability.

The goal of this paper is to show how FIoT can be used to prototype physical smart things based on embodied cognition. Previously, we modeled and simulated smart traffic lights in [6], which were tested in a simulated car traffic application. However, we only provided simulated smart things. Therefore, we did not show how to transfer the evolved controller to physical smart things. For instance, we created a simpler experiment, but we will show all the steps of engineering smart things using evolved neural networks. We will present the steps of modeling and evolving a neural network in a simulated environment, and the step of transferring this evolved network to physical devices.

We present this experiment in Section 4. It presents the experimental setup, results, and evaluation. The remainder of this paper is organized as follows. Section 2 presents the related work. Section 3 describes the background for the proposed approach. The paper ends with conclusive remarks in Section 5.

2 Related Work

There are some research results in the literature about smart things, which use a kind of self-developed approach [15–18]. Baresi et al. [16], for example, provide a simulation for a smart green house scenario. In their simulation, flowers are distributed in different rooms based on specific characteristics. If a flower is sick, it will be allocated to another room or its room's configuration will change. For this purpose, the authors use adaptive techniques to perform discovery, self-configuration, and communication among heterogeneous things. However, most of this research presents only simulated smart things and does not show how to transfer their approaches from a simulated smart thing to a physical one. In [18], one of the few papers that designed a prototype for a smart thing, the authors state that new algorithms need to be integrated to their approach for the development of cooperative smart things. They developed smart street lights that are not able to interact with each other. Thus, each smart street light makes decisions independently.

There are also some commercial applications based on smart things [19–21]. But they do not seem to provide things with a self-developed capability. For example, in HomeKit's [19] approach, the user needs to control and specify the behavior of each one of the smart devices, instead of things having the ability of acting by themselves and learning to adapt. Very recently, IBM proposed the use of the embodied cognition concept in its future products [22, 23] in order to create devices featuring dynamic learning and reasoning about how to act. Their proposed solution is to embed Watson - an IBM platform that uses machine learning techniques, especially neural networks - into smart things.

Besides the software industry starting to discuss the use of embodied cognition to model physical devices, this approach has been used in robotic literature for many years [9, 11, 24–26]. Our goal has been to adapt this approach and show that it is also feasible to model applications based on the Internet of Things that require smart things with self-developed and cooperative capabilities.

3 Background

3.1 Embodied Agents

Embodied agents have a body and are physically situated, that is, they are physical agents interacting not only among themselves but also with the physical environment. They can communicate among themselves and also with human users. Robots, wireless devices and ubiquitous computing are examples of embodied agents [9].

Figure 1 depicts an embodied agent according to the description presented by the Laboratory of Artificial Life and Robotics [27] [28] about embodied agents. They define embodied agents as agents that have a body and are controlled by artificial neural networks [10]. These agents use learning techniques, such as an evolutionary algorithm, to adapt to execute a specific task.

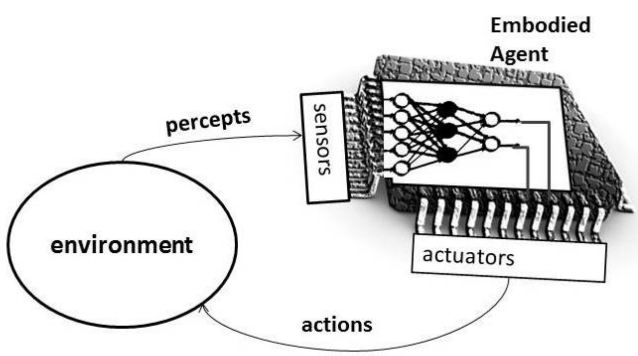

Figura 1: Embodied agent model.

3.2 Evolving Embodied Agents

The authors in [29] describe the process for evolving embodied agents using an evolutionary algorithm, such as genetic algorithm. Accordingly, we provided a simplified flowchart of this process in Figure 2. The interested reader may consult more extensive papers [30–32] or our dissertation [33] (chap. ii, sec. iii).

Normally, the use of an evolutionary algorithm in a multi-agent system provides the emergence of features that were not defined at design-time, such as a communication system. While in traditional agent-based approaches the desired behaviors are accomplished intuitively by the designer, in evolutionary ones these are often the result of an adaptation process that usually involves a larger number of interactions between the agents and the environment [29].

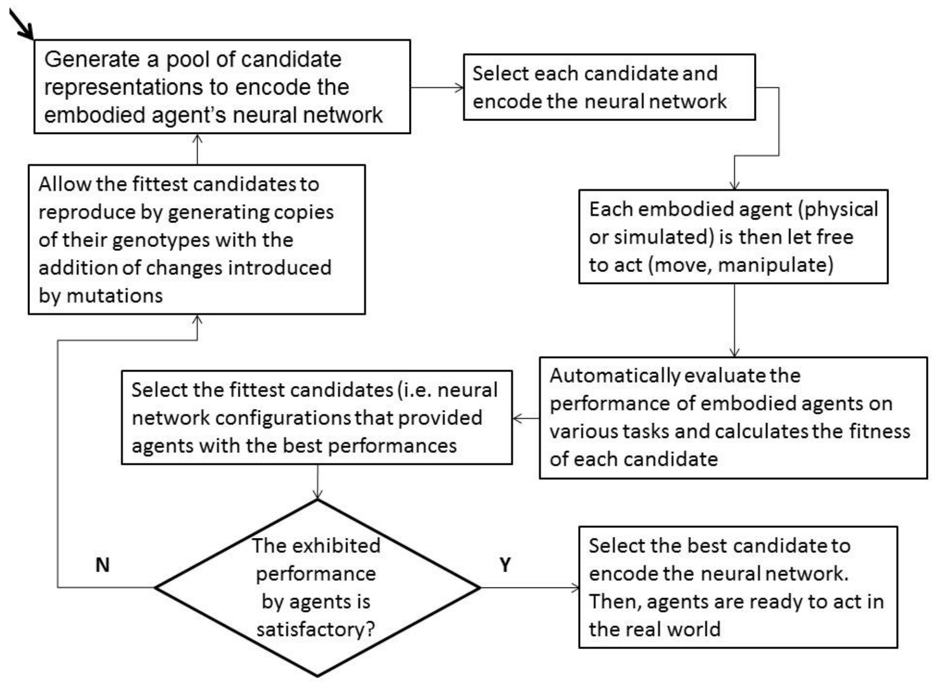

Figura 2: Flowchart: Evolving embodied agents.

The process of evolving an embodied agent's neural network can occur on-line or off-line [34]. The on-line training uses physical devices during the evolutionary process. In such case, an untrained neural network is loaded into a physical agent. Then, the evolution of this neural network occurs based on the evaluation of how this real device behaves in a specific scenario. The off-line training evolves the neural controller in a simulated agent [34], and then transfers the evolved neural network to a physical agent.

The major disadvantage of executing on-line evolution is the time of execution, since evaluating physical devices may require much time. In addition, the training process based on evolution can produce bad configurations for the neural network, which could generate serious problems in particular scenarios. Otherwise, the on-line training insures that evolved controllers function well in real devices.

3.3 FIoT: A Framework for the Internet of Things

The Framework for the Internet of Things (FIoT) [6] is an agent-based software framework that we implemented [6] to generate application controllers for smart things through learning algorithms. The framework does not cover the development of environment simulators, but only the development of smart things' controllers.

If a researcher develops an application using FIoT, his application will contain a Java software already equipped with modules for detecting smart things in an environment, assigning a controller to a particular thing, creating software agents, collecting data from devices and supporting the communication structure among agents and devices.

Some features are variable and may be selected/developed according to the application type, as follows: (i) a control module such as a neural network or finite state machine; (2) an adaptive technique to train the controller; and (iii) an evaluation process to evaluate the behavior of smart things that are making decisions based on the controller. For example, Table 1 summarizes how the "Street Light Control" application will adhere to the proposed framework, while extending the FIoT flexible points.

Tabela 1: FIoT's Flexible Points

FIoT Framework	**Street Light Control Application**
Controller	Three Layer Neural Network
Making Evaluation	Collective Fitness Evaluation: Test a pool of candidates to represent the network parameters. For each candidate, it evaluates the collection of Smart Street Lights, comparing fitness among candidates
Controller Adaptation	Evolutionary Algorithm: Generate a pool of candidates to represent the network parameters

4 Application Scenario: Smart Street Lights

In order to evaluate our proposed approach to create self-developed and cooperative smart things, we developed a smart street light application. The overall goal of this application is to reduce the energy consumption and maintain the maximum visual comfort in illuminated areas. For this purpose, we provided each street light with ambient brightness and motion sensors, and an actuator to control its light intensity. In addition, we also provided street lights with wireless communicators. Therefore, they are able to cooperate with each other in order to establish the most evaluable routes to the passers-by and to achieve the goal of minimizing energy consumption.

We used an evolutionary algorithm to support the design of this system's features automatically. By using a genetic algorithm, we expect that a policy for controlling the street lights, with a simple communication system among them, will emerge from this experiment. Therefore, no system feature such as the effect of ambient brightness on light status changes was specified at design-time.

As we discussed, the training process can occur in a simulated or in a physical environment. However, many devices could be damaged if we were to use real equipment. Therefore, to execute the training algorithm, we decided to simulate how smart street lights behave in a fictitious neighborhood. After the training process, we transferred the evolved neural network to physical devices and observed how they behaved in a real scenario.

4.1 Simulating the environment

In this subsection, we describe a simulated neighborhood scenario, where Figure 3 depicts the elements that are part of the application namely, street lights, people, nodes and edges. We modeled our scenario as a graph, in which a node represents a street light position and an edge represents the smallest distance between two street lights.

Figura 3: Simulated Neighborhood.

The graph representing the street light network consists of 18 nodes and 34 edges. Each node represents a street light. In the graph, the yellow, gray, black and red triangles represent the street light status (ON/DIM/OFF/Broken Lamp). Each edge is two-way and links two nodes. In addition, each edge has a light intensity parameter that is the sum of the environmental light and the brightness from the street lights in its nodes. Our goal is to simulate different lighting in different neighborhood areas.

People walk along different paths starting at random departure points. Their role is to complete their routes, reaching a destination point. A person can only move if his current and next positions are not completely dark. In addition, we also supposed that people walk slowly if the place is partially devoid of light. For simulation purposes, we chose four nodes as departure points (yellow nodes) and two as destinations (red nodes). We started with ten people in this experiment. We also configured that 20% of the street lights lamps would go dark during the simulation.

4.2 Smart Street Light

Each street light in the simulation has a micro-controller that is used to detect the approximation of a person, interact with the closest street light, and control its lights. A street light can change the status of its light to ON, OFF or DIM. Smart street lights have to execute three tasks: data collection, decision- making and action enforcement. The first task consists of receiving data related to people flow, ambient brightness, data from the neighboring street lights and current light status. To make decisions, smart street lights use a three-layer feedforward neural network with a feedback loop [10]. Feedback occurs because one or more of the neural network's outputs influence the next neural network's inputs.

4.3 Creating the Neural Network Controller

We used the FIoT (see Section 3.3) to instantiate the three-layer neural network controller for our smart street lights (see Figures 4 and 5).

```
simplesLightController
<Type>: lightNeuralNetwork
<Input>: 4
previousListeningDecision
lightSensor
motionSensor
wirelessReceiver
<Output>: 3
wirelessTransmitter
listeningDecision
lightDecision
<NHiddens>: 2
<NWeight>: 14
<end>
```

Figura 4: Configuration file to create a neural network controller using FIoT.

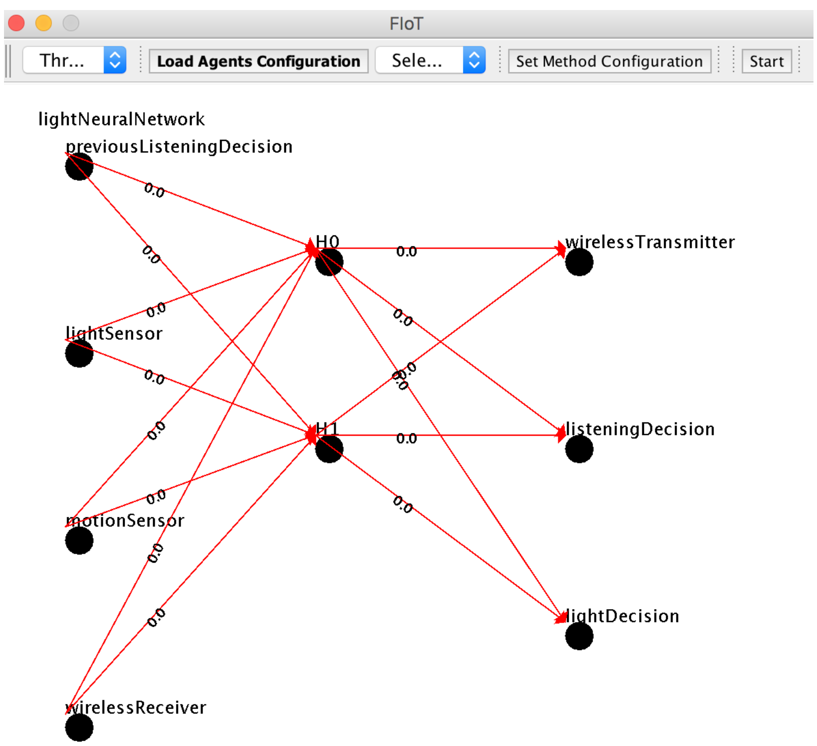

Figura 5: The Neural Network Controller for Smart Street Lights: zeroed weights (FIoT's Application View).

The input layer includes four units that encode the activation level of sensors and the previous output value of listeningDecision. The output layer contains three output units: (i) listeningDecision, that enables the smart lamp to receive signals from neighboring street lights in the next cycle; (ii) wirelessTransmitter, a signal value to be transmitted to neighboring street lights; and (iii) lightDecision, that switches the light's OFF/DIM/ON functions.

The middle layer of the neural network has two neurons connecting the input and output layers. These neurons provide an association between sensors and actuators, which represent the system policies that can change based on the neural network configuration.

4.4 Training the Neural Network

The weights in the neural network used by the smart street lamps vary during the training process, as the system applies a genetic algorithm to find a better solution. Figure 6 depicts the simulation parameters that were used by the evolutionary algorithm. We selected these parameters values (i.e number of generation and tests, population size, mutation rate, etc.) according to known experiments of evolutionary neural networks that we found in the literature [11, 35] (see Figure 2 - Section 3.2).

geneticConfiguration...

```
<numberOfGeneration>:200
<numberOfTests>:20
<numOfBestToBeSelected>:20
<numOfChildren>:4
<numberOfPopulation>:100
<elitismSaveFitness>:1
<maxNumGenes>: 14
<valueMaxOfGene>: 2
<rateMutation>:20
<fitness>:1
```

Figura 6: Configuration file to evolve the neural network via genetic algorithm using FIoT.

During the training process, the algorithm evaluates the weight possibilities based on the energy consumption, the number of people that finished their routes after the simulation ends, and the total time spent by people to move during their trip. Therefore, each weights set trial is evaluated after the simulation ends based on the following equations:

$$pPeople = \frac{(completedPeople \times 100)}{totalPeople} \tag{1}$$

$$pEnergy = \frac{(totalEnergy \times 100)}{\left(\frac{11 \times (timeSimulation \times totalSmartLights)}{10}\right)} \tag{2}$$

$$pTrip = \frac{(totalTimeTrip \times 100)}{\left(\left(\frac{3 \times timeSimulation}{(2)}\right) \times totalPeople\right)} \tag{3}$$

$$fitness = (1.0 \times pPeople) - (0.6 \times pTrip) - (0.4 \times pEnergy) \tag{4}$$

in which *pPeople* is the percentage of the number of people that completed their routes as of the end of the simulation out of the total number of people in the simulation; *pEnergy* is the percentage of energy that was consumed by street lights out of the maximum energy value that could be consumed during the simulation. We also considered the use of the wireless transmitter to calculate energy consumption; $pTrip$ is the percentage of the total duration time of people's trips out of the maximum time value that their trip could spend; and $fitness$ is the fitness of each representation candidate that encodes the neural network.

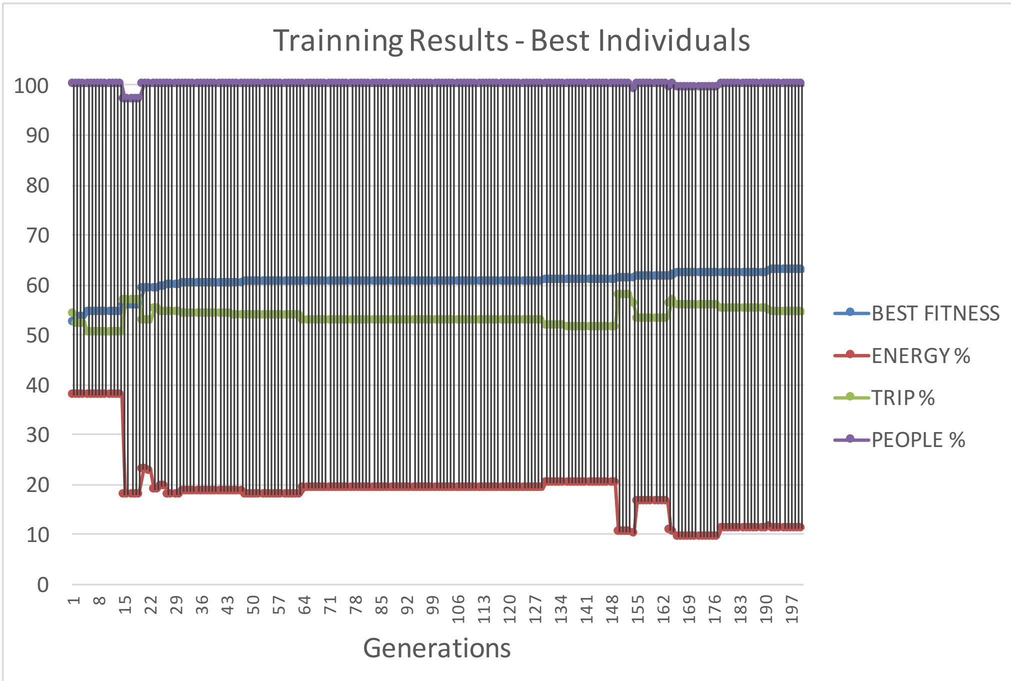

Figura 7: Simulation results - Most-Fit from each generation.

Normally, the performance of the most-fit individual is better than the others. Figure 7 illustrates the best individual from each generation (i.e. the candidate with the highest fitness value). As shown, the best individuals from the generations have tend to minimize energy consumption and find an equilibrium between energy consumption and the trip time. We selected the best individual from the last generation to investigate its solution, as shown in the subsection below (4.4.1).

4.4.1 Evaluation of the Best Candidate

After the end of the evolutionary process, the algorithm selects the set of weights with the highest fitness (equation 4). Figure 8 depicts the evolved neural network configured with the best set of weights found during the evolution.

One disadvantage of using neural networks combined with evolutionary algorithms is to understand and explain the behaviors that were automatically assigned by the smart things. Therefore, we executed the simulated street lights using the evolved network in order to generate logs and extract the rules that are implicit in patterns of the generated input-output mapping. For example, Figure 9 depicts some logs that were generated by using the runtime monitoring platform proposed by Nascimento et al. [36] and that we analyzed to understand why and when street lights decided to communicate and switch the lights ON.

After analyzing logs, we could realize the rules that were created by the evolved neural network in order to understand why and when street lights decided to communicate and switch the lights ON. The code below exemplifies some of these rules:

$$\begin{aligned}(I_0 = 1.0 \wedge I_1 = 0.5 \wedge I_2 = 0.0 \wedge I_3 = 0.0) \Rightarrow \\ (Out_0 = 0.0 \wedge Out_1 = 1.0 \wedge Out_2 = 0.0)\end{aligned} \tag{5}$$

$$\begin{aligned}(I_0 = 1.0 \wedge I_1 = 0.5 \wedge I_2 = 1.0 \wedge I_3 = 0.0) \Rightarrow \\ (Out_0 = 0.0 \wedge Out_1 = 1.0 \wedge Out_2 = 0.5)\end{aligned} \tag{6}$$

$$\begin{aligned}(I_0 = 0.0 \wedge I_1 = 0.0 \wedge I_2 = 0.0 \wedge I_3 = 0.0) \Rightarrow \\ (Out_0 = 0.5 \wedge Out_1 = 0.0 \wedge Out_2 = 0.0)\end{aligned} \tag{7}$$

$$\begin{aligned}(I_0 = 1.0 \wedge I_1 = 0.0 \wedge I_2 = 0.0 \wedge I_3 = 0.5) \Rightarrow \\ (Out_0 = 0.0 \wedge Out_1 = 1.0 \wedge Out_2 = 0.5)\end{aligned} \tag{8}$$

in which the variables are:

$$\begin{aligned}I_0 \equiv previousListeningDecision, I_1 \equiv lightSensor, \\ I_2 \equiv motionSensor, I_3 \equiv wirelessReceiver, \\ O_0 \equiv wirelessTransmitter, O_1 \equiv listeningDecision, \\ O_2 \equiv lightDecision\end{aligned} \tag{9}$$

Based on the generated rules and the system execution, we could observe that only the street lights with broken lamps emit “0.5"by its wireless transmitter (rule 7). In addition, we also observed that a street light that is not broken switches its lamp ON if it detects a person's approximation (rule 6) or receives “0.5"from wireless receiver (rule 8) .

Discussion

Imagine if we had to codify into the physical smart lights all of these rules that could be operated by this evolved neural network. Using the evolved neural network, we saved lines of code and programming time. The code size is an important parameter in this kind of project, since it is normally composed of devices with many resource constraints.

We provided street lights with the possibility of disabling the feature of receiving signals from neighboring street lights. In an initial instance, we did not consider broken lamps. Therefore, as the action of communication increases energy consumption, the street lights decided to disable this feature. However, when we added broken lamps to the scenario, during the evolutionary process, the solution of enabling a communication system among street lights provided better results. Therefore, as shown in the rules generated by the evolved neural network, a smart street light takes lightSensor, motionSensor and wirelessReceived inputs into account to make decisions. Thus, the best solution does not ignore any of these parameters.

One advantage of engineering physical devices based on embodied cognition is that the found solution normally is sufficiently generic. To estimate how generic is the approach, we simulated another neighborhood with a different number of street lights and a different configuration map, then we applied this best solution to this new scenario. The results showed that the evolved street lights' behavior do not vary based on the number of street lights, and the lighting application continues functioning well even if we disable some street lights in the scenario.

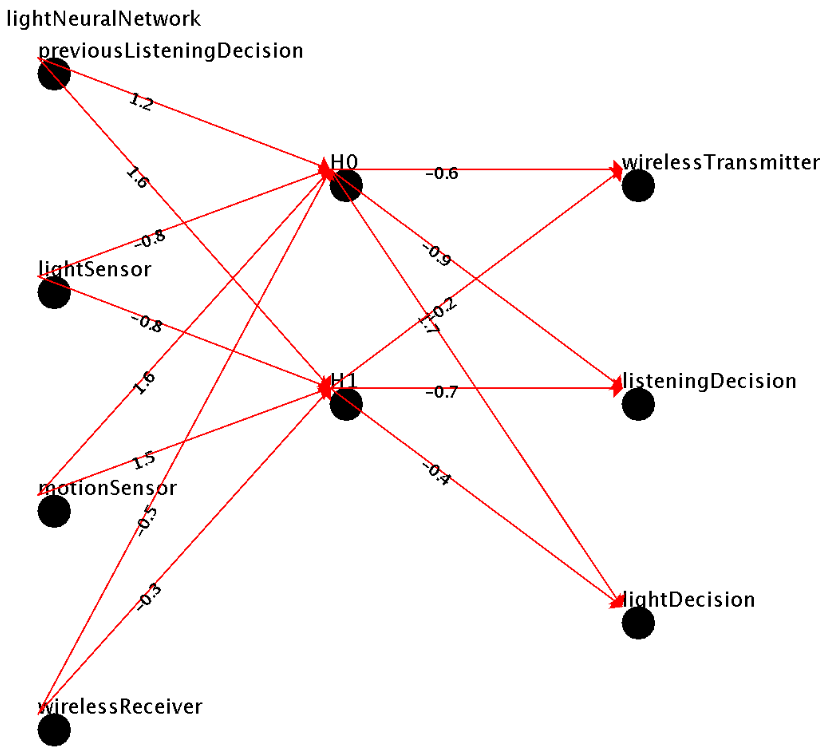

Figura 8: The Evolved Neural Network to be used as a controller for real Street Lights (FIoT's Application View).

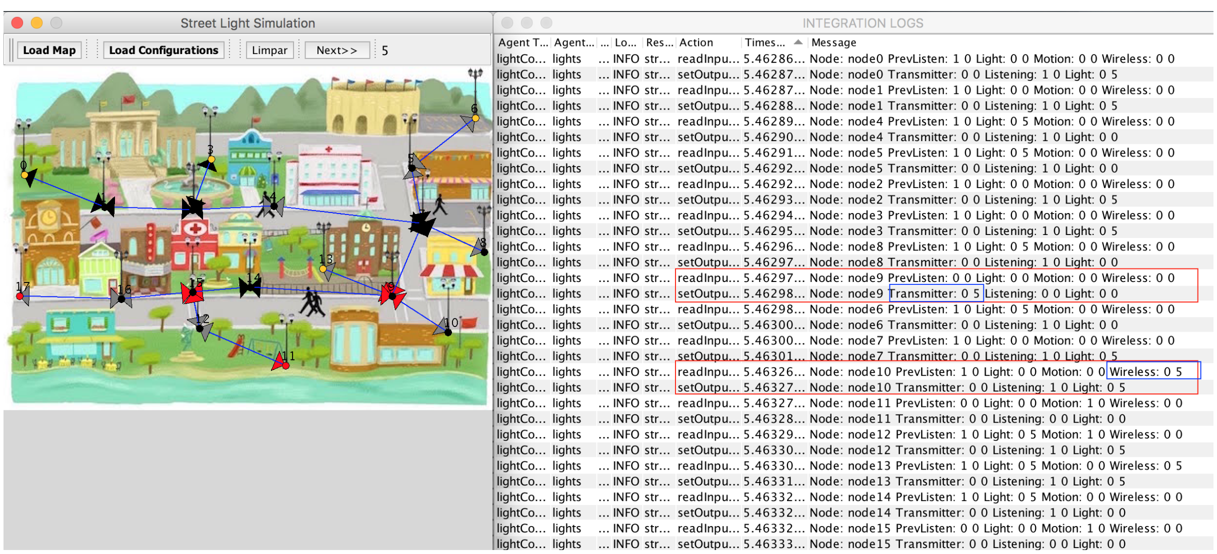

Figura 9: Log Analysis: Why and when evolved street lights switch the lights ON or emit the signal "0.5"?

4.5 Prototyping the Smart Street Light Device

As depicted in Figure 10, the prototype of the Smart Street Light is composed of an Arduino [37] and the following sensors and actuators: (i) HC-SR501 (a device that detects moving objects, particularly people. The detection distance is slightly shorter - maximum of 7 meters); LM393 light sensor (a device to detect the ambient brightness and light intensity); nRF24L01 (a wireless module to allow one device to communicate with another); and (iii) LEDS (the representation of a lamp).

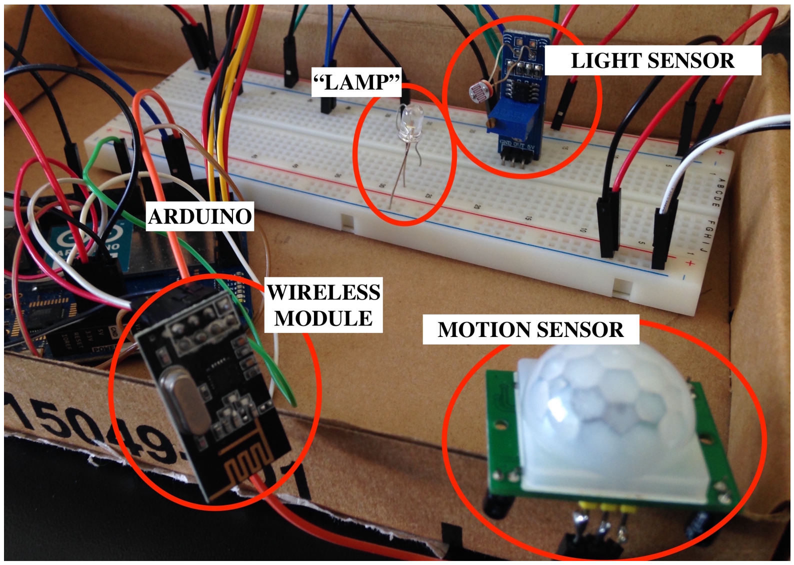

Figura 10: Prototyping the Smart Street Light.

We put two LEDs in this circuit. Our goal is to simulate light intensity. Therefore, if a smart street light decides to set its light intensity to the maximum, both LEDs will be on. If the light intensity is medium, one LED will be on and the other LED will be off.

4.6 Transferring the evolved neural network to physical devices

After the neural network has been evolved, we codified it into the Arduino. We show below the code in C++ language that operates as a neural network inside the Arduino:

```
double fSigmoide(double x){
double output = 1 / (1 + exp(x));
return output;
}
```

```
double calculateHiddenUnitOutput(double w[4]){
double H = previousListeningDecision*w[0] +
lightSensor*w[1]+motionSensor*w[2]+wirelessReceiver*w[3];
double HOutput = fSigmoide(H);
return HOutput;
}
```

```
double calculateOutputDecisions(double w[2], double h0, double h1){
double outputSum = h0*w[0] + h1*w[1];
double output = fSigmoide(outputSum);
return output;
}
```

As we described in section 4.2, each smart street light has to execute three tasks. Accordingly, we present below the main parts of the C++ code that the Arduino executes to attend to the tasks of collecting data, making decisions and enforcing actions:

- Collecting data:

```
void getInputs(){
lightSensor = readLightSensor();
motionSensor = readMotionSensor();
previousListeningDecision = listeningDecision;
if ( listeningDecision ==1){
receivedSignal = receiveWirelessData();
}
else
receivedSignal = 0;
}
```

- Making Decision (calculating output decisions based on the evolved neural network functions):

```
double weightsH0[4] = 1.2, -0.8, 1.6, -0.5;
double weightsH1[4] = 1.6, -0.8, 1.5, -0.3;
double H0 = calculateHiddenUnitOutput(weightsH0);
double H1 = calculateHiddenUnitOutput(weightsH1);

...
double weightsTransmitterOutput[2] = -0.6, -0.2 ;
double transmitterOutput = calculateOutputDecisions(weightsTransmitterOutput, H0, H1);

...
double weightslisteningDecision [2] = -0.9, -0.7;
double listeningDecisionOutput = calculateOutputDecisions(weightslisteningDecision, H0, H1);

...
double weightslightDecision [2] = 1.7, -0.4;
double lightDecisionOutput = calculateOutputDecisions(weightslightDecision, H0, H1);
if (lightDecisionOutput>threshold2){
lightDecision = 1.0;
}
else{
if (lightDecisionOutput>threshold1){
lightDecision = 0.5;
}
else lightDecision = 0.0;
}
```

- Enforcing action:

```
void setOutputs(){
 ...
sendWirelessData(transmitterSignal);
 ...
writeLed(lightDecision);
 ...
}
void writeLed(double value){
 if (value == 1){
 digitalWrite(ledPin, HIGH);
 digitalWrite(led2Pin, HIGH);
 }
 else if(value == 0.5){
 digitalWrite(ledPin, HIGH);
 digitalWrite(led2Pin, LOW);
 }
 else{
 digitalWrite(ledPin, LOW);
 digitalWrite(led2Pin, LOW);
 }
 }
```

4.7 Testing Physical Smart Street Lights in a Real Scenario

In a controlled real scenario, we put three prototypes of the smart street lights using the evolved neural network into operation. We distributed them in the scenario as shown in Figure 11.

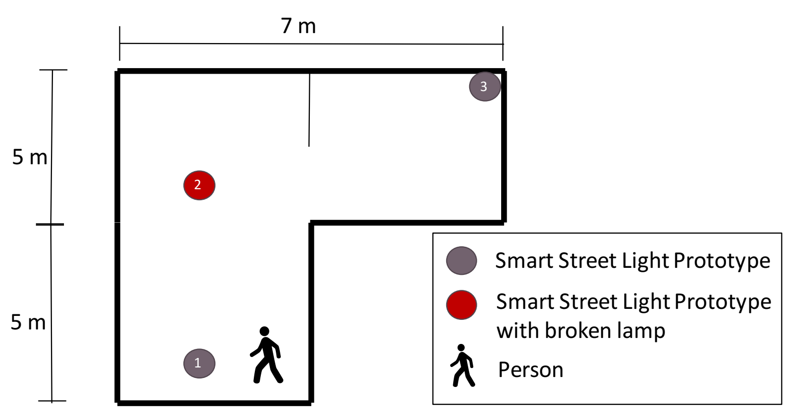

Figura 11: Real Scenario where we tested a network of three smart street lights prototypes.

To compare the behavior of physical smart street lights to the simulated ones, we also collected logs from the Arduinos. As we could observe, the behavior of the physical smart street lights was similar to the simulated ones: it switches lamps ON if it receives a signal from the closest street light or detects the approximation of a person. The major difference occurred because we used LEDs, instead of real lamps. Therefore, the brightness of LEDs did not interfere on the light sensor as did a real lamp [1].

5 Conclusion and Future Work

We believe these preliminary results are promising. We proposed the use of the embodied cognition concept to model smart things. To illustrate, we modeled and implemented smart street lights. Each smart street light had sensors and actuators to interact to the environment, and used an artificial neural network as a internal controller. In addition, we used a genetic algorithm to allow smart street lights to self-develop their own behaviors through a non-supervised training. As a result, a group of initially non-communicating smart street lights developed a simple communication system. By communicating, the group of street lights seems to cooperate in order to achieve collective targets. For example, to maintain the maximum visual comfort in illuminated areas, the street lights used communication to reduce the impact of broken lamps.

After evolving the neural controller, we designed three homogeneous prototypes of the smart street light and transferred the evolved controller into their microcontrollers. We put them in a real scenario and compared them to the simulated street lights. Previously, we described, in [6], a more complex application, but we only had provided a simulated scenario. In this work, we showed that is possible to automatically create and train a smart thing's controllers using FIoT and to use it to control physical smart things.

As an ongoing work, we need to improve the real scenario, testing the use of the evolved network to control real street lights in a real neighborhood. In addition, we need to develop more realistic scenarios, taking several other environmental parameters into account. Furthermore, since we had shown that the use of an evolved neural network results in saving code lines, we also need to test this experiment using microcontrollers with fewer resources, such as battery and memory. Another challenge from creating more realistic scenarios is to model heterogeneous experiments, training different smart things in the same scenario. For example, the application of smart waste collecting will require two types of smart things: smart trash cans and smart waste collection vehicles. Therefore, these different types of smart things will need to cooperate with each other in order to achieve the goal of minimizing waste transportation costs and promoting environmental sustainability.

Our next goal is to allow the system to initiate a new learning process after the evolved network has been already transferred to the physical smart things. Therefore, we will change the neural network's parameters at run-time and allow the real smart things to adapt their behavior in the face of changing environmental demands. For this purpose, we need to use a simulator for wireless devices that allow our training system to communicate with and for programming microcontrollers at runtime, such as Terra [38], which is a system for programming wireless sensor network applications. Therefore, our system will evaluate physical smart things' behaviors at runtime, execute adaptation in a more realistic simulated environment via a learning algorithm, and then automatically transfer the trained controller to the physical smart things.

[1] All files that were generated during the development of this work, such as genetic algorithm files, log program and arduino code, are available at
http://www.inf.puc-rio.br/˜ nnascimento/streetlight.html

6 Acknowledgment

This work has been supported by the Laboratory of Software Engineering (LES) at PUC-Rio. Our thanks to CNPq, CAPES, FAPERJ and PUC-Rio for their support through scholarships and fellowships.

7 References

[1] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[2] J. O. Kephart, "Research challenges of autonomic computing," in *Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on.* IEEE, 2005, pp. 15–22.

[3] P. Horn, "Autonomic computing: Ibm\'s perspective on the state of information technology," IBM, Tech. Rep., 2001.

[4] HP, "Adaptive enterprise: Infrastructure and management solutions for the adaptive enterprise." Hewlett-Packard Development Company, Tech. Rep., 2003.

[5] Microsoft, "Microsoft dynamic systems initiative overview." Microsoft, Tech. Rep., 2004.

[6] N. M. do Nascimento and C. J. P. de Lucena, "Fiot: An agent-based framework for self-adaptive and self-organizing applications based on the internet of things," *Information Sciences*, vol. 378, pp. 161–176, 2017.

[7] M. E. Markiewicz and C. J. P. de Lucena, "Object oriented framework development," *Crossroads*, vol. 7, no. 4, pp. 3–9, Jul. 2001.

[8] K. Cetnarowicz, K. Kisiel-Dorohinicki, and E. Nawarecki, "The application of evolution process in multi-agent world to the prediction system," in *Second International Conference on Multiagent Systems*, 1996, pp. 26–32.

[9] L. Steels, "Ecagents: Embodied and communicating agents," SONY, Tech. Rep., 2004.

[10] S. Haykin, *Neural Networks: A Comprehensive Foundation.* Macmillan, 1994.

[11] D. Marocco and S. Nolfi, "Emergence of communication in embodied agents evolved for the ability to solve a collective navigation problem," *Connection Science*, 2007.

[12] S. Nolfi and D. Parisi, "Learning to adapt to changing environments in evolving neural networks," in *Adaptive Behavior*, 1997, pp. 75–98.

[13] B. Jacob, R. Lanyon-Hogg, D. K. Nadgir, and A. F. Yassin, "A practical guide to the ibm autonomic computing toolkit," 2004.

[14] D. Floreano and C. Mattiussi, *Bio-Inspired Artificial Intelligence. Theories, Methods, and Technologies.* Cambridge: MIT Press, 2008.

[15] A. Katasonov, O. Kaykova, O. Khriyenko, S. Nikitin, and V. Y. Terziyan, "Smart semantic middleware for the internet of things." *ICINCO-ICSO*, vol. 8, pp. 169–178, 2008.

[16] L. Baresi, S. Guinea, and A. Shahzada, "Short paper: Harmonizing heterogeneous components in sesame," in *Internet of Things (WF-IoT), 2014 IEEE World Forum on.* IEEE, 2014, pp. 197–198.

[17] L. Zhu, H. Cai, and L. Jiang, "Minson: A business process self-adaptive framework for smart office based on multi-agent," in *e-Business Engineering (ICEBE), 2014 IEEE 11th International Conference on.* IEEE, 2014, pp. 31–37.

[18] J. F. De Paz, J. Bajo, S. Rodríguez, G. Villarrubia, and J. M. Corchado, "Intelligent system for lighting control in smart cities," *Information Sciences*, vol. 372, pp. 241–255, 2016.

[19] Apple, "Homekit," https://developer.apple.com/homekit/, March 2017.

[20] Samsung, "Samsung smart things," https://www.smartthings.com, March 2017.

[21] Postcapes, "Tracking the internet of things," http://postscapes.com/connected-home, November.

[22] IBM, "Built with watson: Embodying project intu," https://www.ibm.com/blogs/watson/2017/01/embodying-project-intu/, January 2017.

[23] ——, "Built with watson: Experiment with embodied cognition with project intu," https://www.ibm.com/blogs/watson/2016/11/experiment-embodied-cognition-project-intu/, November 2016.

[24] A. Loula, R. Gudwin, C. N. El-Hani, and J. Queiroz, "Emergence of self-organized symbol-based communication in artificial creatures," *Cognitive Systems Research*, vol. 11, no. 2, pp. 131–147, 2010.

[25] D. Floreano, S. Mitri, S. Magnenat, and L. Keller, "Evolutionary conditions for the emergence of communication in robots," *Current biology*, vol. 17, no. 6, pp. 514–519, 2007.

[26] S. Nolfi, J. Bongard, P. Husbands, and D. Floreano, *Evolutionary Robotics.* Cham: Springer International Publishing, 2016, ch. 76, pp. 2035–2068.

[27] D. Parisi and S. Nolfi, "Laboratory of autonomous robotics and artificial life," website, 1994.

[28] S. Nolfi, "Laboratory of autonomous robotics and artificial life," LARAL, http://laral.istc.cnr.it/, Tech. Rep., March 1995.

[29] S. Nolfi and D. Floreano, *Evolutionary Robotics: The Biology,Intelligence,and Technology of Self-Organizing Machines.* Cambridge, MA, USA: MIT Press, 2000.

[30] G. F. Miller, P. M. Todd, and S. U. Hegde, "Designing neural networks using genetic algorithms," in *Proceedings of the third international conference on Genetic algorithms.* Morgan Kaufmann Publishers Inc., 1989, pp. 379–384.

[31] R. K. Belew, J. McInerney, and N. N. Schraudolph, "Evolving networks: Using the genetic algorithm with connectionist learning," in *In.* Citeseer, 1990.

[32] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.

[33] N. M. Nascimento, "FIoT: An agent-based framework for self-adaptive and self-organizing internet of things applications," Master's thesis, PUC-Rio, Rio de Janeiro, Brazil, August 2015.

[34] A. Nelson, G. Barlow, and L. Doitsidis, "Fitness functions in evolutionary robotics: A survey and analysis," *Robotics and Autonomous Systems*, 2007.

[35] M. DA VIDE and S. Nolfi, "Emergence of communication in teams of embodied and situated agents," in *The Evolution of Language: Proceedings of the 6th International Conference (EVOLANG6), Rome, Italy, 12-15 April 2006.* World Scientific, 2006, p. 198.

[36] N. Nascimento, C. J. Viana, A. v. Staa, and C. J. P. Lucena, "A publish-subscribe based approach for testing multi-agent systems," *Monografias em Ciência da Computação*, vol. 2016, p. 21, 2016.

[37] Arduino, "Arduino," http://www.arduino.cc/, February 2017.

[38] A. Branco, F. Sant'anna, R. Ierusalimschy, N. Rodriguez, and S. Rossetto, "Terra: Flexibility and safety in wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 11, no. 4, p. 59, 2015.