A semi-automatic multi agent intelligent system for production processes

Candidate: Nunzio Turco

Supervisors: Fulvio Corno, Luigi De Russis

Company tutor: Rosaria Rossini - Links Foundation -

Introduction

The rise of **Industry 4.0** has led factories to focus on connectivity and interactions among machines and people. This new technology enables faster, more flexible and more efficient processes to produce higher-quality goods at reduced costs. In this context, the **multi agent system** ability to communicate finds a relevant role. Nowadays, Multi agent systems are used in a wide range of applications, such as in industry optimization, game playing, market simulation, especially in case of complex scenario, where a distributed approach can help to simplify the problem. But in most of these applications, their intelligence is limited only on the communication ability, this is especially true in case of industrial scenario, where due to the low risk policy, a complete automation of the process is venturesome and risky.

Thesis Objective

The objective of the thesis is to design, develop and test a multi-agent intelligent system in the context of industry 4.0, aiming at improving different aspects of the production process, such as anomaly detection, advise actions and facilitate monitoring of the machines. This system, called **NTMAS**, wants to provide a partial automation of the process exploiting a **Human-in-the-loop model**, in this way the human supervisor has a complete control on what is happening on the plant. Each agent action must be validated by a supervisor before it takes effect, this is also a way to improve its decisions. Moreover, it comes with a web **dashboard** that allows user to interact with the agents and monitor the system.

The NTMAS will be tested in a partially simulated environment, derived from a real dataset. In this way it will be possible to analyse the behaviour of the agents, in particular their ability to communicate and to take decision autonomously. At the end, an evaluation of the NTMAS advantages and limits is performed.

System design

The NTMAS is divided in two main parts, one is the **agents ecosystem** and the other one is the **visualization and storage component**. The first part is composed of all the agents gathering

information about the machines on which they are deployed, an Agent Management System (AMS) keeping track of all the active agents and a White Pages service storing this information; these last two components are required to make the system architecture **FIPA** compliant. The second part is formed by a server which contains a database with all the data collected by the agents and runs a web application for visualizing the current machines status and allowing user to interact with the system. NTMAS contains two different types of agents, besides AMS, one is the Producer and the other is the Inspector. This latter is thought to be deployed on quality check machines, since it has the role of analysing the data it is perceiving and raising alerts in case of production problems. The Producer instead is in charge of deciding who and what kind of action to take in case of alarm. First, each producer generates an error vector, measuring how much the currents machines values have changed with respect to the past one and sends it to the other agents to select the faulty one; then it exploits a pre-trained online machine learning model to select an action from a given set of possibilities. But, before the action takes place, it must be validated or changed by the user, so this agent sends a request to the visualization and storage component, which allows the user to have the last word. From that response the agent will also update the machine learning model, as to be able to adapt better to the scenario.

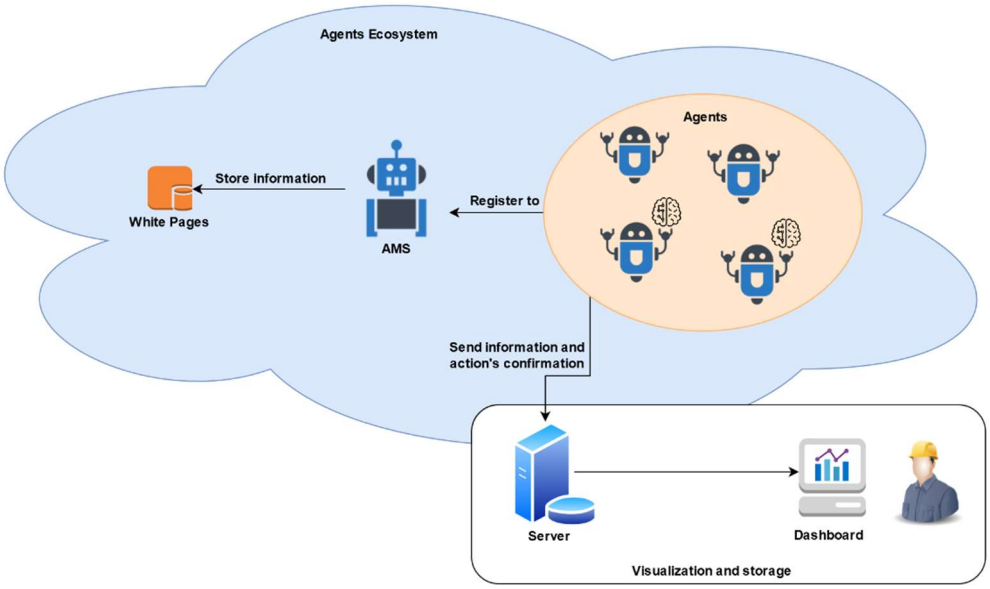

Figure 1: System architecture

Development

NTMAS is written in **Python**, and each component, besides the server and the dashboard, is deployed inside a **Docker** container. Inside the agent ecosystem, the entities communicate using a **RabbitMQ** broker, which exploits AMQP message protocol. The White-Pages service contains a database based on **MySQL** with all the online agents information and only the AMS has access to it through MySQL connector. Inspector and Producer have different sub-goals but the same objective, they both run a Flask service to communicate with the server and to allow programmers to interact directly with them while they are running. Moreover, the production agent includes a **LSTM network** to select an action in case of alert; the choice of this machine learning method is motivated by the fact that it allows to keep training the algorithm while the system is in execution. All the deployed agents send the gathered information to a server running a **Flask service**, this server stores the data and provides it to a client running a **JavaScript** code that plots this information inside multiple graphs. Moreover, the client allows the user to interact with the Production agents by means of buttons that appear in case of action request, as can be seen from the image below.

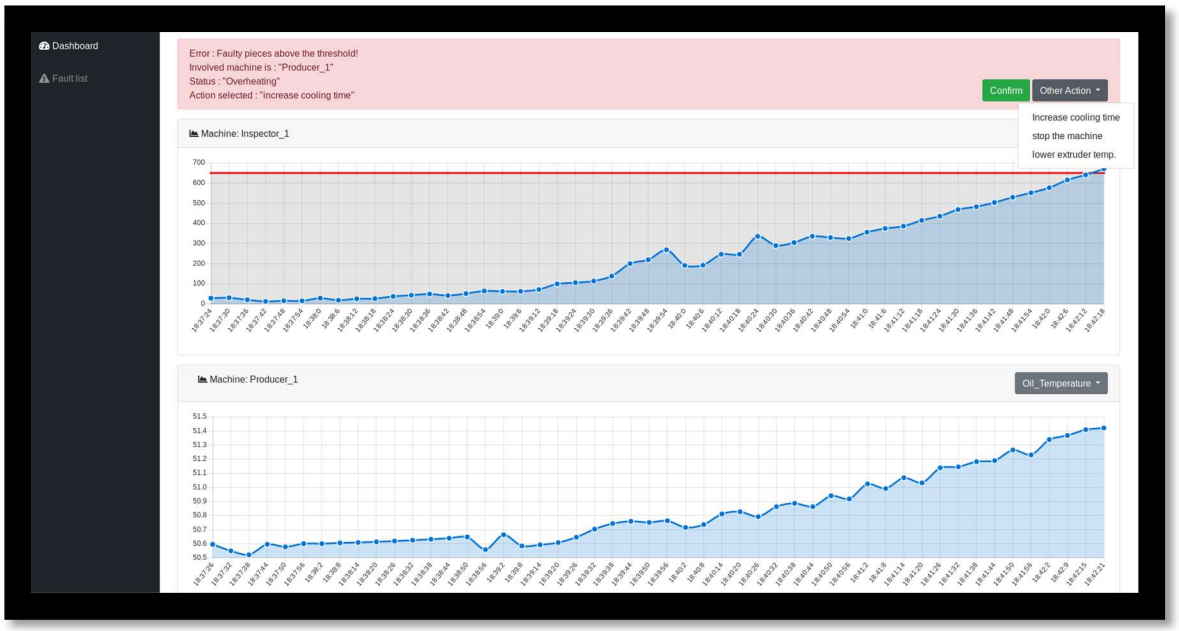

Figure 2: Dashboard

Evaluation and results

The evaluation starts from the study of a use case concerning a real injection molding plant, which data has been provided by Links Foundation. Since this dataset contains anomalies caused by security checks not labelled, and also because during these inspections some parameters may be tuned and/or some components substituted, it was impossible to understand the natural behaviour of the system. Hence, it was decided to **simulate four types of errors**:

- Overheating
- Loss of pressure
- Stuck of plastic part
- Increase of melt temperature.

Moreover, to each error a corresponding action was proposed. This new dataset has been first divided into training and test set, then the LSTM network was trained on the first set and its model with all the weights was saved into a file that was uploaded inside the production agents. Two types of tests have been made, each one requiring a different number of agents: one with **1 inspector and 1 producer** and another with **1 inspector and 3 producers.** In both cases the NTMAS correctly handled the situation and the system has adapted correctly to the introduction of more agents. In particular two kinds of collaboration ability arise: the first one between inspector and producer, for the understanding of a problem inside the production process, and the second between multiple producers, for electing the faulty machine.

Conclusions and Future Work

Even if the NTMAS didn't show any problem in the simulated scenario, some limits are presents:

- It is not possible to add a new action without re-train the entire machine learning model.
- The system could not handle multiple alerts at the same time: if during the user decision another alert rises, the previous one is discarded.

Besides removing these limits, future research should aim to deploy the proposed system in a real production environment to test its effectiveness.