

Data filtering model based on Self-Assembly and Quorum-Sensing in bacterial colonies

Giacomo Mantani

Alma Mater Studiorum – University of Bologna
via Sacchi 3, 47522 Cesena, Italy
giacomo.mantani@studio.unibo.it

Abstract. Stromatolites are colonies of bacteria that self-assemble into rock formations in tidal salt flats. Each stromatolite can make independent decisions, while maintaining communication with the colony. The workload is shared among all colony individuals. Stromatolites breed rapidly, and quickly develop resistance to antibiotics and other threats by developing new genes. I plan to incorporate these principles into a simulated data filtering system. The system is composed by autonomous agents acting like bacteria. Workload i.e., incoming data, will be spread evenly through the network. And in a process mimicking natural selection, desirable data will be quickly distributed to incoming knowledge, while undesirable data will be filtered.

1 Introduction

For many years, bacterial cells are considered primarily as selfish individuals, but, in recent years, it has become evident that, far from operating in isolation, they coordinate collective behavior in response to environmental challenges using sophisticated inter cellular communication networks. Cell-to-cell communication between bacteria is mediated by small diffusible signal molecules that trigger changes in gene expression in response to fluctuations in population density. This process, generally referred to as quorum sensing (**Quorum Sensing**). Recent advances have revealed that bacteria are not limited to communication within their own species but are capable of **listening in** and **broadcasting to** between also unrelated species (*heterogeneous system*). This *behavior* allow them to intercept messages and coerce cohabitants into behavioral modifications.

2 Background

In most environments, bacteria are found in mixed communities, and therefore populations of a given **Quorum Sensing** species that can sense a signal broadcast information both to themselves and to other species sharing the same ecological niche (*Environ*).

Different bacterial species may *speak* the same **Quorum Sensing language**, some may possess sensors for specific **Quorum Sensing** signals that facilitate eavesdropping while others may manipulate the **Quorum Sensing** activities of neighboring bacteria by degrading the **Quorum Sensing** signal molecules within their locality. See Figure 1.

Bacteria are the most abundant and adaptive form of life on Earth and, as

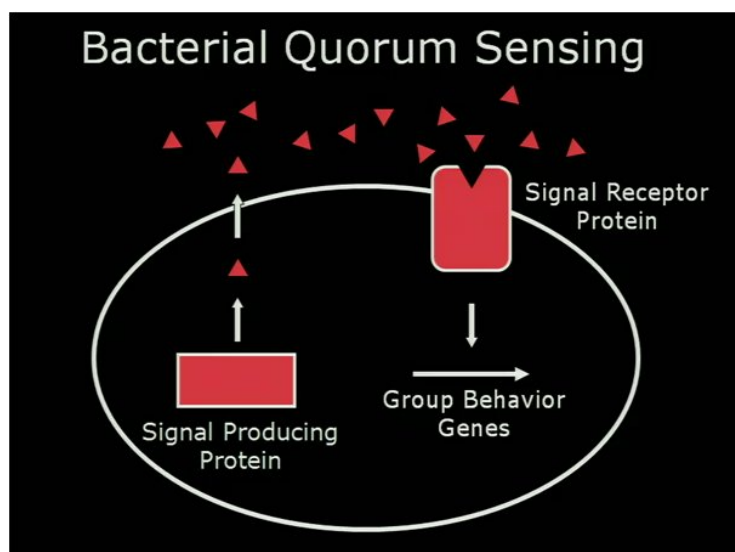


Fig. 1. Releasing molecules = Quorum Sensing [Bas09]

such, have an impact on the habitats and lifestyles of all other living organisms. We would therefore be naive to think that, over the millennia, coevolution of pro- and eukaryotes has not stimulated the development of inter-kingdom signalling pathways promoting parasitic, symbiotic or commensal relationships; the members of each biological kingdom possess *hormone-like* molecules responsible for cell-cell communication, a fact that is fundamental to the concept that the members of any given kingdom will respond to the signals produced by another. See Figure 2.

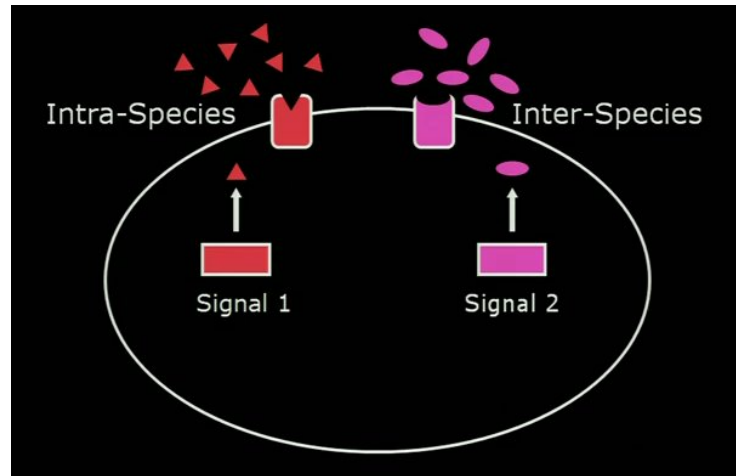


Fig. 2. Communication Channels [Bas09]

Bacteria, in general, are capable of **coordinating** their behavior collectively. **Quorum Sensing** is now a well-established concept in bacteriology, is increasingly being discovered in unicellular eukaryotic microbes and has even been used to describe the behavior of ants [PMSF02], metastatic cells and even iron-impregnated particles [SPT⁺09]. There is also the intriguing possibility that different species from one kingdom may coordinate their behavior against the individual/s from another kingdom. Irrespective of how the signals are processed, the results will either be a collective change in behavior for the good of the community as a whole or will favor one or more species over others.

For further information, Bonnie's talk at **TED** has more insights about "*How bacteria 'talk'*" [Bas09].

3 Project Analysis

The project goal is to reproduce/emulate bacteria behavior learned in previous section. The simulation shall prove that using computational agents that mimicking a nature inspired system could help us make progress on data-filtering problems like for example, **SPAM**. Before going deeper into the problem solution, I will define some notions.

We [Omi15] define *simulation* as:

- "the imitation of the operation of a real-world process or system over time" [Par01]
- "Simulation is the process with which we can study the dynamic evolution of a model system, usually through computational tools" [Ban99]

Simulation requires a **model**, a representation/abstraction of an actual system that involves AGGREGATION, SIMPLIFICATION and OMISSION.

Citing Siddhartha Mukherjee [Muk15] *"You know, when an architect builds a model, he or she is trying to show you a world in miniature. But when a scientist is building a model, he or she is trying to show you the world in metaphor. He or she is trying to create a new way of seeing. The former is a scale shift. The latter is a perceptual shift."*

In our example simulation create a virtual laboratory where generation of an artificial evolution in bacterial colonies came up. We need a model that should be able to capture:

- the presence of **autonomous entities** (*bacterium*) whose actions and interactions determine the evolution of the system
- effects of local-global interaction, self-organization and emergence of proprieties related only to the whole system

We know that such a model exist and it is called **Agent-based Model**. We [Omi15] have defined it as *"a specific individual-based computational model for studying macro emergent phenomena through the definition of the system micro level which is modeled as a collection of interacting entities."*

Now we try to explain our building blocks in terms of:

- Structure
- Interaction
- Behavior

3.1 Structure

Starting with the **environment**. It is composed by different connected area. Each area has its own:

1. α Bacteria
2. β bad data
3. Γ good data
4. Δ other *intra*-species molecules that help each bacteria filter *bad* data

Good and *bad* data are metaphors of nearby **Quorum Sensing** molecules in real life scenario. These molecules could influence bacteria lives.

Bacterium as living system has a *boundary* and an *immune system* wired in his genes. The boundary purpose is [Omi15]:

- set it apart from what it is not
- simultaneously regulate its interactions with the environment

Bacteria are not heteronymous because *input* and *output* alone do not determine the system. Bacteria like all other living entities establish their own cycles in **time**. *Live, do something, die.*

3.2 Interaction

The interaction between bacteria is mediated by the environment. Each bacterium releases molecules outside his boundary and it perceives *intra*-species and *inter*-species ones with receptors. *There is not point-to-point communication between two bacteria.* In order to achieve data filtering goals only *intra*-species communication is important. These two channel communication could lead into a **security mechanism** (one channel could be work like a trusted one and the another not).

3.3 Behavior

Bacterium act like a "*dumb*" organism that do only these things:

1. Perceive other *intra*-species bacteria in the area
2. If the perceived number is greater than a threshold start **Quorum Sensing** phase:
 - Perceive molecules/**immune genes** from others *intra*-species bacteria
 - Release outside his boundary (into the *environment*) its own molecules/**immune genes**
3. Else move to another nearby area
4. Filter outside molecules that are not from trusted source (it could be DANGEROUS). If an unknown molecule is "*bad*" and bacterium has not immune gene that could destroy the molecule it will be damaged. Otherwise it decompose the molecule and other *intra*-species bacteria will benefit from this action (in the same environment).

Autonomy characteristics captured into the whole system are:

- Boundaries, each bacterium and environment area
- **Homeostasis**, separating internal molecules knowledge from external ones
- Internalization, thanks to preceptors bacteria are able to internalize other bacteria knowledge (molecules) released into the environment
- Gain in size, in terms of *immune genes* that are able to filter more *bad* data
- Flexibility, bacteria prefer areas with high intra-species population density in order to make stromatolites but they could also survive alone

More information about autonomy and which resources to change it could be found in [Ros14].

4 Project Implementation

The implementation could be made using different technologies and frameworks. I choose TuCSoN [AP1c] in order to achieve our goal for two reasons:

- It is a good model that can handle the problem without abstraction gaps
- We [Omi15] have used it in previous activities and so I have working examples to begin with.

4.1 Structure

Before we begin with implementation details, I would like to say something about the main directories and file in the root project:

- *README.rst*, a classical README that you can find in every **Version Control System**
- *bac*, source code root directory
- *background*, science papers about bacteria, **Quorum Sensing**, etc.
- *paper*, L^AT_EX root directory, with all files needed to generate this report

Now follow an in depth description of project's source-code **structure**. Inside BAC I have *libs* directory with needed libraries (**tucson.jar** and **2p.jar**) and *src* directory with the package SA.MANTANI.TUCSON. Inside the package:

- BACTERIA:
 - **Bacteria**, release a configurable random number of bacterium in each *Area (TupleCentre)*
 - **Bacterium**, **AbstractTucsonAgent** that mimic real life behavior discussed in previous sections **2 3**
 - **Gene**, each bacterium has his own immune system that help him **filter** *bad* data perceived in the **Environment**
 - **ImmuneSystem**, set of genes in a bacterium
- DATA:
 - **Data**, in order to decouple future filtering rules, it is an **Adapter**
 - **DataInfo**, knowledge base for **Data**
- ENV:
 - **Area**, class used by the bacterium that has informations about near areas (*TupleCentre*)
 - **Environment**, init each *TupleCentre* defined in the *Topology* with *good* and *bad* data
- GUI:
 - **SwarmListener**, listener for *button* press and *timer* tick actions
 - **SwarmMonitor**, TuCSoN *TupleCentre* monitor to update the **GUI**
 - **bacteriaGUI**, base class to start the **GUI**
 - **bacteriaGUI.form**, *XML* file with form meta-data
- LAUNCHERS:
 - **LaunchEnvironment**, self explain
 - **LaunchGUI**, self explain
 - **LaunchScenario**, start everything, **main**
 - **LaunchTopology**, self explain
- UTILS:
 - **Boot**, start TuCSoN Node
 - **Installer**, init TuCSoN Node
 - **Topology**, *TupleCentres* topology configuration

4.2 Interaction

All interaction process are governed by TuCSoN *TupleCentres*. Each *TupleCentre* has a finite set of tuples. This implementation has the following tuples' structure (*Prolog* like):

- Environment's Data, `data(N)` like `data(-15)` (*bad*) and `data(25)` (*good*)
- Bacterium, `bacterium(NAME,HEALTH)` like `bacterium(top0,42)`
- Near *TupleCentres*, `nbr(TCID)` like `nbr('@'(center,':(localhost,20504)))`

4.3 Behavior

Implemented application is a simulation program that let you tune parameters in order to achieve different outputs. Configurable parameters will be inside a configuration file in a future release, but at the moment you can find them inside the following class like *static final* variables:

- Inside `DataInfo`:

```
public abstract class DataInfo {
    public static final int BAD_DATA_LIMIT = 25;
    public static final int GOOD_DATA_LIMIT = 25;
    public static final int BAD_DATA_EACH_AREA = 15;
    public static final int GOOD_DATA_EACH_AREA = 25;
    ...
}
```

- Inside `Bacterium`:

```
public class Bacterium extends AbstractTucsonAgent {
    private final static Long TIMEOUT = 500L;
    private final static int NGENES = 3;
    private final static int STROMATOLITE_THRESHOLD = 5;
    private final static int HEALTH = 50;
    private final static int MAX_CYCLES = 500;
    ...
}
```

- Inside `Bacterium`:

```
public class Bacterium extends AbstractTucsonAgent {
    private final static Long TIMEOUT = 500L;
    private final static int NGENES = 3;
    private final static int STROMATOLITE_THRESHOLD = 5;
    private final static int HEALTH = 50;
    private final static int MAX_CYCLES = 500;
    ...
}
```

- Inside Bacteria:

```
public final class Bacteria {  
  
    private final static int BACTERIA_PER_AREA = 20;  
  
    ...  
  
}
```

- Topology is handle at runtime by the system, you could change it and always should work fine. NB **GUI** is hardcoded only for Topology:

```
public final class Topology {  
    public static final List<Area> areas = new  
        → ArrayList<>(5);  
  
    public static void bootTopology() {  
        try {  
            →Area center = new Area("center", "localhost",  
            → "20504");  
            Area top = new Area("top", "localhost", "20512");  
            Area right = new Area("right", "localhost", "20503");  
            →Area bottom = new Area("bottom", "localhost",  
            → "20506");  
            Area left = new Area("left", "localhost", "20509");  
  
            ...  
        }  
    }  
}
```

5 Conclusion

Running `LaunchScenario`, **GUI** looks like 3. In the screenshot, we can see in a poorly way how bacteria survive better in team. Stromatolite that came up in *center TupleCentre* has more **filter** rules that help bacteria inside have better **health average**. This is an emergent behavior thanks to *intra-species Quorum Sensing* bacteria interaction. This simulator could be used and adapted in a more sophisticated **anti-spam** scenario, where *bad* data has not a simple structure like a negative integer number. It is only required to adapt `Data` class.

Last but no the least I would like to thank Andrea Omicini, Stefano Mariani and all collaborators to TuCSoN eco-system, without whom this paper would be less than few words of theory.

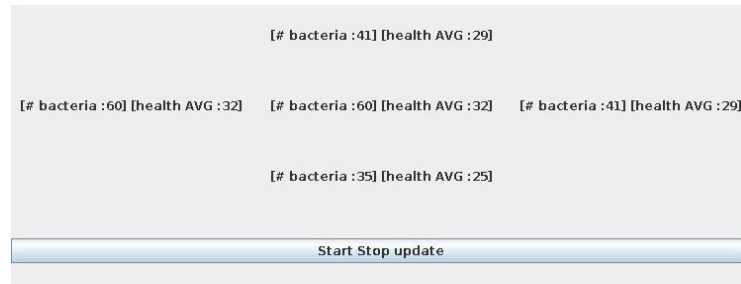


Fig. 3. Working Example Screenshot

6 Information about the author



Glossary

Agent-based Model An agent-based model (*ABM*) is a specific individual-based computational model for studying macro emergent phenomena through the definition of the system micro level which is modeled as a collection of interacting entities . 4, 9

GUI Graphical User Interface. 6, 8, 9

Homeostasis Ability of a system to regulate its internal conditions to keep some or several functions stable. 5, 9

Quorum Sensing Cell-to-cell communication between bacteria mediated by small diffusible signal molecules that trigger changes in gene expression in response to fluctuations in population density. 1–6, 8, 9

SPAM Unsolicited or undesired electronic messages.. 3, 9

TED TED is a platform for ideas worth spreading. Started in 1984 as a conference where technology, entertainment and design converged, TED today shares ideas from a broad spectrum — from science to business to global issues — in more than 100 languages. Meanwhile, independent TEDx events help share ideas in communities around the world. 3, 9

TuCSon TuCSon[APIC] (Tuple Centres Spread over the Network) is a Java library providing coordination as a service to Java and tuProlog[APIId] agents. TuCSon is a model for the coordination of distributed processes, as well as autonomous, intelligent and mobile agents. TuCSon exploits tuple centres as its coordination media, which are tuple spaces enhanced with the notion of behaviour specification, thanks to the ReSpecT[APIb] language. 9

Version Control System also known as revision control or source control, is the management of changes to documents, computer programs, large web sites, and other collections of information. Changes are usually identified by a number or letter code, termed the "revision number," "revision level," or simply "revision". An example is GIT. 6, 9

References

- APIa. APICe. Lab. <http://apice.unibo.it>.
- APIb. APICe. Reaction specification tuples. <http://apice.unibo.it/xwiki/bin/view/ReSpecT/>.
- APIC. APICe. Tuple centres spread over the network. <http://apice.unibo.it/xwiki/bin/view/TuCSon/>.
- APIId. APICe. tuprolog, light-weight prolog system for distributed applications and infrastructures. <http://apice.unibo.it/xwiki/bin/view/Tuprolog/>.

- AW09. Steve Atkinson and Paul Williams. Quorum sensing and social networking in the microbial world. *Journal of The Royal Society Interface*, 2009.
- Ban99. Jerry Banks. Introduction to simulation. In *Proceedings of the 31st conference on Winter simulation: Simulation—a bridge to the future-Volume 1*, pages 7–13. ACM, 1999.
- Bas09. Bonnie Bassler. How bacteria "talk", February 2009.
- Muk15. Siddhartha Mukherjee. Soon we'll cure diseases with a cell, not a pill. https://www.ted.com/talks/siddhartha_mukherjee_soon_we_ll_cure_diseases_with_a_cell_not_a_pill, March 2015.
- Omi15. Andrea Omicini. Sistemi autonomi (2015-2016). <https://apice.unibo.it/xwiki/bin/view/Courses/Sa1516>, 2015.
- Par01. Domenico Parisi. *Simulazioni: la realtà rifatta nel computer*. Il mulino, 2001.
- PMSF02. Stephen C Pratt, Eamonn B Mallon, David J Sumpter, and Nigel R Franks. Quorum sensing, recruitment, and collective decision-making during colony emigration by the ant *leptothorax albipennis*. *Behavioral Ecology and Sociobiology*, 52(2):117–127, 2002.
- Ros14. Bernd Rosslenbroich. *On the origin of autonomy: A new look at the major transitions in evolution*, volume 5. Springer Science & Business Media, 2014.
- SPT⁺09. Grant S Stewart, Stephanie Panier, Kelly Townsend, Abdallah K Al-Hakim, Nadine K Kolas, Edward S Miller, Shinichiro Nakada, Jarkko Ylanko, Signe Olivarius, Megan Mendez, et al. The riddle syndrome protein mediates a ubiquitin-dependent signaling cascade at sites of dna damage. *Cell*, 136(3):420–434, 2009.