On Exploiting Decentralized Bio-inspired Self-organization Algorithms to Develop Real Systems

Elisabetta Di Nitto, Daniel J. Dubois, Raffaela Mirandola

Dipartimento di Elettronica e Informazione
Politecnico di Milano

SEAMS 2009 – Vancouver, Canada – May 18th, 2009
Objectives

• **Problem**: increasing complexity of computing systems.

• **Possible solution**: autonomic computing [IBM manifesto].
  – Superimpose on a software system a control layer (the MAPE cycle)

• **Our Goal**:
  – A more decentralized solution based on **self-organizing logic**
    • use analogies taken from the biological world to accomplish complex tasks using simple rules.
  – Suitable for pervasive systems often obtained as combination of pre-existing components
Self-Organization

The spontaneous evolution of a system into an organized form in the absence of external pressures.

• Self-organization algorithms are able to guide the evolution of the system toward a specific goal with no or minimal user intervention.

• Example: Vineyard control system.
Example: Vineyard control system
Bio-inspired Self-Organization: forms of self-organization that come from observed phenomena of the natural world.

- Examples from the natural world: finding the optimal path in ant colonies, firefly synchronization, etc.

• Advantages
  – Scalability,
  – Robustness,
  – Malicious or malfunctioning components,
  – …

• Disadvantages
  – We miss proper design approaches supporting their application.
What are the ingredients for applying self-organization to a software system?

• Self-organization principles.

• Algorithms.
  – May combine bio-inspired principles.

• Development guidelines
  – List of steps to be executed to move from the model of algorithms/principles to the actual implementation.
Bio-inspired Principles

• Derived from the observation of natural phenomena.

• They may be composed together and translated into real deployable algorithms.

• Most important principles:
  – Noise
  – Emergence
  – Diffusion
  – Stigmergy
  – Evolution
Noise Principle

“a strategy of errors”

Definition: the use of some kind of perturbations to move a system away from its expected goal in the short term, but increasing the probability to improve it in the long term.
Emergence Principle

"a team is better than the sum of its individuals"

Definition: capability of a system composed by multiple components to reach (with a certain probability) a global goal by achieving local goals at component level.
Communication principles: Diffusion and Stigmergy

**Diffusion**

“*gossiping is not necessarily a bad habit*”

Definition: a method of disseminating information among many interconnected components. Messages are sent to all known nodes (flooding) or random nodes (gossiping).

**Stigmergy**

“*leave a note and I’ll read it*”

Definition: a method of communication that stores information in the environment.
Evolution Principle

Definition: capability of the system to improve itself using a natural selection process among its components.

• Best components tend to survive, worst components tend to die.
Stochasticity Property

- Bio-inspired principles are expressed in a probabilistic way.
- They may increase the probability that the system reaches a desired state, but they do not assure that the system actually reaches it.
- Our question is no longer whether systems work, but what is the probability that they work.

- All properties of the system (functional and non-functional) are expressed in stochastic terms:
  - system keeps working in presence of unexpected behaviors
  - absence of guarantees on when a specific goal is reached.
Example of algorithm: Self-LoadBalancing

Goal: balance the workload in a network of interconnected heterogeneous nodes.

- Consider a set of nodes;
- Nodes are connected in an overlay network;
Example of algorithm: Self-LoadBalancing

Goal: balance the workload in a network of interconnected heterogeneous nodes.

- Consider a set of nodes;
- Nodes are connected in an overlay network;
- Each node belongs to a type;
Example of algorithm: Self-LoadBalancing

Goal: balance the workload in a network of interconnected heterogeneous nodes.

- Consider a set of nodes;
- Nodes are connected in an overlay network;
- Each node belongs to a type;
- Each node has a queue of jobs of its type;
Example of algorithm: Self-LoadBalancing

Goal: balance the workload in a network of interconnected heterogeneous nodes.

- Consider a set of nodes;
- Nodes are connected in an overlay network;
- Each node belongs to a type;
- Each node has a queue of jobs of its type;
- Each node may receive jobs from the environment or from neighbor nodes;
Example of algorithm: Self-LoadBalancing

Goal: balance the workload in a network of interconnected heterogeneous nodes.

- Consider a set of nodes;
- Nodes are connected in an overlay network;
- Each node belongs to a type;
- Each node has a queue of jobs of its type;
- Each node may receive jobs from the environment or from neighbor nodes.

- No centralized control;
- No global knowledge over the network;
- Each node may interact with its neighbors only.
Example of algorithm: Self-LoadBalancing

Goal: balance the workload in a network of interconnected heterogeneous nodes.

- Consider a set of nodes;
- Nodes are connected in an overlay network;
- Each node belongs to a type;
- Each node has a queue of jobs of its type;
- Each node may receive jobs from the environment or from neighbor nodes.

- No centralized control;
- No global knowledge over the network;
- Each node may interact with its neighbors only.
Bio-inspired Self-Aggregation

Iterative rewiring among small groups of 3 nodes leads to the **emergence** of the self-aggregated network.

Local optima are avoided using the **noise** principle.
Dimension Exchange Load Balancing

The Dimension Exchange Load Balancing algorithm balances the workload of each couple of nodes until all the nodes have the same load (global balancing emerges after multiple local balancings).
Self-Aggregation + DE Load Balancing [bio08]

1. Group nodes of the same type into homogeneous domains using a self-aggregation algorithm;

2. Balance the workload among homogeneous domains using the Dimension Exchange Load Balancing algorithm.

Implementation Issues

• The proposed model does not capture:
  – Conflicts when handling concurrent iterations.
  – Activation/Termination problem.
  – Choice of interval between algorithm iterations.

• Possible solutions:
  – Locking mechanisms.
  – Keeping the algorithm always active VS finding a propagation strategy.
  – Heuristics for automatically determining iteration intervals.

• Stochasticity:
  – No guarantees that the optimal solution is obtained, however the probability may be estimated.
Creating Self-org algorithms for Software Systems

- Model often does not capture the following problems:
  - Components synchronization.
  - Race conditions.
  - Algorithm initialization and termination.
  - Frequency of algorithm iterations.
  - Amount of network communication.
  - Alternative strategies for dealing with stochasticity.
Conclusions

• We explore the problem of applying bio-inspired self-organization techniques to real systems.
• It is important to reason in terms of how algorithms/principles are applied in real deployable systems.

• Future Work:
  – A complete systematic methodology consisting in:
    • a list of steps and design patterns based on bio-inspired principles.
    • heuristics that can be used to calibrate the algorithms.
  – Identify suitable validation approaches.
QUESTIONS?
Self-Optimization in P/S systems

1. **Training phase**: evaluate interests by measuring traffic for each subject.

2. **Reconfiguration**: choose the best rewiring based on an utility function that uses the information gathered during the training phase.
Simulation Environment

- Distributed Simulator written in Java.
- 20 Monte Carlo Simulations.
- Network of 100 nodes, average node degree of 4 neighbors.
- Scale-Free topology.
- Heterogeneity: 10% (10 types of nodes/jobs).
- Job distribution: static load of 400 jobs and continuous insertion of 400 jobs every 20s.
- Node processing time: a) 100% 5s, b) 70% 7s, 30% 3s (ideal throughput 20jobs/s).
- Node churn: every 10s 20% of the nodes disappears, and the same number of new nodes appears.

We evaluate the throughput and the network load in terms of message overhead.
Simulations without Rewiring

**Number of processed jobs without rewiring**

Load balancing iterations are inhibited by the fact that the jobs cannot traverse the nodes having different types.
Simulations with Rewiring: static load

**Number of processed jobs with rewiring**
We have a strong improvement with respect to the previous experiments
Simulations with Rewiring: multiple bursts

Number of processed jobs with rewiring and multiple bursts
The curve grows as a straight line because of the continuous job bursts that are sent to the nodes.
Simulations with Rewiring: multiple bursts

Throughput of processed jobs with rewiring and multiple bursts
The throughput is close to the optimal value (the one obtained in homogeneous networks)
Simulations: multiple bursts and churn

Throughput of processed jobs in presence of a node churn (arrival/departure) of 20%/10secs. The throughput is similar to the previous case, therefore it is resistant to this type of uncertainty.
Simulations: different service time

Throughput of processed jobs in presence of nodes of different processing times
Like in the previous case the combined algorithm is still able to obtain nearly-optimal values.
Message overhead (in terms of number of messages) grows linearly with time in all simulations since it is dominated by the self-aggregation algorithm (5 msgs/sec vs 0.033 msgs/sec).
Summary

• Strengths:
  – Throughput is close to 20 in all experiments, meaning that the load is spread among almost all workers;
  – Network overhead can be controlled and adapted to the application scenario;
  – System shows self-healing property in presence of churn and dynamism.
Summary

• **Strengths:**
  – Throughput is close to 20 in all experiments, meaning that the load is spread among almost all workers;
  – Network overhead can be controlled and adapted to the application scenario;
  – System shows self-healing property in presence of churn and dynamism.

• **A possible problem:**
  – Churn may remove key nodes from the network (nodes with high degree) and split the network;
  – However churn and dynamism, especially arrival of new nodes makes this “problem” transitory.