TEK9010 - Exam prep summary

Sebastian G. Winther-Larsen

December 1, 2020

0: General concepts in multiagent systems

Agents

Could you describe an agent?

An agent isa computer system capable of independent action on behalf of its owner or user. Agents need skills and abilities to cooperate, coordinate and negotiate with each other on behalf of their users.

We deal with two different kinds of agents,

1. Reactive (or simple) agents can produce emergent properties usually modelled by Swarm Intelligence (SI).
2. Strategic (or intelligent) agents engage in social activities like cooperation, coordination, negotiation, etc usually described with Game Theory (GT)

In some situations an agent-based solution would be appropriate.

1. The environment is open, or at least highly dynamic, uncertain or complex. In these settings autonomous agents might be the only solution.
2. Agents are natural metaphors for socities, organisations and businesses, as well as intelligent interfaces such as "expert assistants".
3. In istribution of data, control or expertise. When centralised solutions are difficult, like synchronisation of many autonomous databases.
4. When dealing with legacy software, one can wrap the legacy software in an agent layer.

Multiagents systems

What is a multiagent system?

Multiagent systems consists of many agents interacting with each other through some network or sensor system.

Multiagent systems is an appropriate software paradigm for modelling and building massive open and complex distributed systems. It is also a natural metaphor for artificial social systems. The research goal of multiagents systems is to connect behaviour on the microscopic scale with (often) emergent properties and effects on the macroscopic scale.

There are some key challenges associated with multiagent system desing.

1. The agent design problem on the microscopic level. How do we build agents that are capable of independent, autonomous action in order to successfully carry out tasks that we delegate to them?
2. The social desgin problem on the macroscopic level. How do we build agents that are capable of intercating with other agents in order to successfully carry out the tasks that we delegate to them, especially when the agents do no share common goals or intentions?

A truly succesful multiagent system makes an explicit connection between the autonomous micro level agents and the macro level modelling of the complex system.

Mutliagent system applications can be divided into two groups,

1. Distributed systems, where agents are processing nodes in a distributed multiagent system (emphasis on "multi").
2. Personal software assistants, where individual agents act as proactive assistants to users (emphasis on "individual").

Swarm intelligence

How would you define swarm intelligence?

Stigmergy

What is stigmergy?

Emergence

What do we mean by emergence in SI?

9: Swarm Robotics 1

What is swarm robotics?

Quick answer: swarm intelligence applied to robotics.

There is no explicit definition of a *swarm* in literature. A swarm is defined via its behavior.

The *size of a swarm* is defined by what it is not: "not as large as to be dealt with statistical averages" and "not as small as to be dealt with as a few-body problem". The size of a swarm N is

$$10^2 < N << 10^23,$$

not Avagadro-large.

Swarm robotics is “the study of how a large number of relatively simple physically embodied agents can be designed such that a desired collective behavior emerges from local interactions among agents and between agents and the environment”, according to Dorigo and Sahin. But! A swarm is not necessarily

There are some key features. The fact that local interactions between agents and the environment should be possible requires robots to have local sensing and probably also communication capabilities. In fact, (local) communication is considered a key feature of swarms.

Collaboration is required to go beyond a mere paralllisation in swarm a swarm system. We want to go beyond the performance of simple parallelisation. Think of some clearning task with each robot cleaning a small assigned area.

Swarm performance.

Some keywords are *contention* or *inference* and (lack of) *coherency*, given by parameters α and β, repectively. The robots need to share limited resources and communicate. This is difficult.

In the contexst of swarm robotics we can interpret contention as interference between robots due to shared resouces, such as an entrance to a base station or generally space. Collision avoidance is a waiting loop because the shared resource *space* is currently not available. This can be compared to an airplane flying in a holding pattern because the resource “runway” is currently in use and should certainly not be shared. Incoherency, in turn, can be interpreted as inconsistencies or overhead due to limited communication of imformation or due to imperfect synchrony.

The univeral scalability law is important,

$$R(N) = \frac{N}{1 + \alpha(N-1) + \beta N(N-1)}.$$

Its inverntor, Gunther, identifies four qualitatively different situations,

1. If contention and lack of coherency are negligible, then we get “equal bang for the buck” and have a linear speedup ($\alpha = 0$, $\beta = 0$),
2. If there is a cost for sharing resources in the form of contention, then we have a sublinear speedup ($\alpha > 0$, $\beta = 0$),
3. If ther is an increased negative influence due to contention, then the speedup clearly levels off ($\alpha >> 0$, $\beta = 0$).
4. If in additon there is also an increased influence of incoherence, then there exists a peak speed up and for bigger system sizes the speedup decreases ($\alpha >> 0$, $\beta > 0$).

One can identify some “regions” of performance; super-linear, sub-linear, optimal and inference. As more agents are added, performance starts to increase (sub-

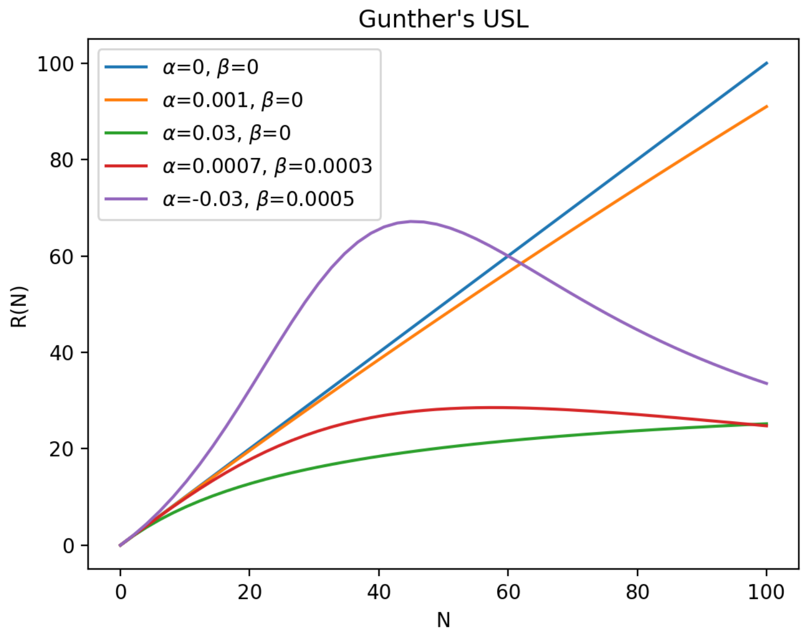

Figure 1: Gunther's Universal law of Computational Scalability

/super-)linearly, then we get to an optimum after a while. After that comes the inference region.

In parallel computing, superlinear speedups can occur due to some interplay between problem size per computing unit and available memory. For example, if the problem can be divied into pieces that fit completely into a CPUs cache, then one can observe a considerable speedup. In swarm robotics, superlinear performance increases occur due to qualitatively different collaboration modes that are accessible with increasing swarm size as seen in bucket brigades.

It is possible for a system-wide deadlock to occur in a swarm robotics system. For instance with a very high swarm denisty, such that all robots permanently try to avoid collisons resulting in zero performance.

Modelling swarms (as a series of mappings).

A swarm system of size N in 2D space can be described by the state vecotr,

$$\gamma = (r_1, r_2, \ldots, r_N, v_1, v_2, \ldots, v_N, s_1, s_2, \ldots, s_N),$$

wher r_i are the positions, v_i are the velocities and s_i are the discrete states for agents $i \in [1, N]$. We denote the configuration space by Γ, i.e. $\gamma \in \Gamma$, with $\dim \Gamma = 2N + 2N + N = 5N$.

This is a large space to keep track off, and we can only observe one sequence, $\gamma_t, \gamma_{t+1}, \gamma_{t+2}, \ldots$ at a time for a specific initial state γ_0. What we ideally need instead is to understand how the system operates in gemeral for any setup. We need to omit certain parts, s.t. we obtain a different definition of a configuration, $\phi \in \Phi$ with $\dim \Phi << \dim \Gamma$. We seek a mapping,

$$f : \Gamma \mapsto \Phi.$$

In a discrete time model, we also need two update rules, $g : \Gamma \mapsto \Gamma$ and $h : \Phi \mapsto \Phi$. We now have the following functionality,

$$f(\gamma_t) = \phi_t,\ g(\gamma_t) = \gamma_{t+1},\ h(\phi_t) = \phi_{t+1}.$$

The following requirement should hold (we want this),

$$h(f(\gamma_t)) = f(g(\gamma_t)),$$

that is, the modelling abstractions implemented by f should be chosen carefulle sucht that the model update rule h is able to predict the correct model configuration for the next time step.

We would also like an inverse map $f^{-1} : \Phi \mapsto \Gamma$ that reverses the model abstractions and rebuilds the configuration γ of the real system from the model configuration ϕ. However, typically f^{-1} cannot usefully be defined, because f is surjective, that is, we can have $\gamma_1, \gamma_2 \in \Gamma$ with $\gamma_1 \neq \gamma_2$ and $f(\gamma_1) = f(\gamma_2)$ due to the reduction in dimensionality caused by f.

Why do we need models in Swarm Robotics? Quote from Schweitzer:

> To gain insight into the interplay between microscopic interactions and macroscopicfeatures, it is important to find a level of description that, on the one hand, considers specificfeatures of the system and is suitable for reflecting the origination of new qualities, but, onthe other hand, is not flooded with microscopic details. (...) A commonly accepted theory of agent systems that also allows analytical investigations is, however, still pending because of the diversity of the various models invented for particular applications.

An extreme example of an extreme model abstraction,

$$f(\gamma) = \frac{|\{s_i | s_i = A\}|}{N} = \phi,$$

where $\dim \Phi = 1$. How does the update rule h look like? Non-trivial!

In swarm robotics we are still on the search for an appropriate general modelling technique.

When are rate equations appropriate?

Rate equations are appropriate when there is no special information of interest in the full state of the system. We are only interested in macroeconomics properties This is typically when we are working with concentrations, for example. Such systems are typically inspired by chemical systems.

The Langevin equation.

$$\dot{\mathbf{R}}(t) = \mathbf{A}(\mathbf{R}(t), t) + B(\mathbf{R}(t), t)\mathbf{F}(t),$$

where $\mathbf{R}$ is the position of an agent, $\mathbf{F}$ is a stochastic process (e.g. white noise), $\mathbf{A}$ describes and scales the agent's behaviour and B describes and scales non-deterministic behaviour. A possible choice for $\mathbf{A}$ is a gradient descent in a potential field,

$$\mathbf{A}(\mathbf{R}(t), t) = \nabla P(\mathbf{R}(t), t).$$

The Focker-Planck equation.

The ~ is a PDE describing the temporal dynamics of a probability density. This density describes in the original physical conetext the probability of finding the particle within a certain area. It is the macroscopically corresponding piece to the microscopic approach described by the Langevin equation. It was originannly used in physics for modelling Brownian motion with drift, describing diffusion processes in thermodynamics.

So, it looks like this,

$$\frac{\partial \rho(\mathbf{r}, t)}{\partial t} = \nabla(\mathbf{A}(\mathbf{r}, t)\rho(\mathbf{r}, t)) + \frac{1}{2}Q\nabla^2(B^2(\mathbf{r}, t)\rho(\mathbf{r}, r)),$$

where ρ is a (probability) density for a single particle at position $\mathbf{r}$ and time t. We interpret it as the robot density of all coexisiting robots of the swarm. By integrating over an area W,

$$s(t) = \int_{\mathbf{r} \in W} \rho(\mathbf{r}, t),$$

we get the expected fration for the swarm within that area at time t. The first term in the Fokker-Planck equation is a non-stochastic drift term and the second term is a diffusion term.