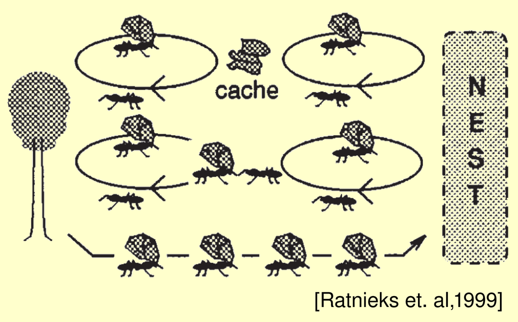
SOS-TC: Organisation and Control of Self-Organising Systems in Technical Compounds

Alexander Scheidler, Daniel Merkle and Martin Middendorf

Parallel Computing and Complex Systems Group, University of Leipzig, Germany, {scheidler,merkle,middendorf}@informatik.uni-leipzig.de

Design and Analysis of Ant Queue Inspired Task Partitioning Methods

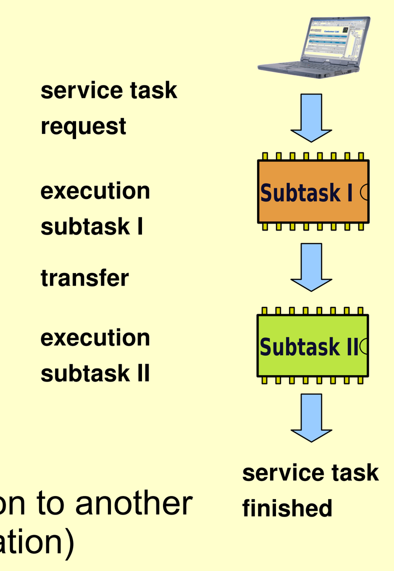


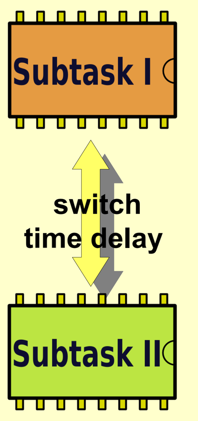
Social Insects

- can divide tasks into subtasks
- worker can specialize to different subtasks
- overall system performance increased (depends on colony size)

Ant Queue inspired Task Partitioning

- service tasks are divided into subtasks
- every component can specialize to one type of subtask (reconfigurable hardware)
- transfer of tasks between components



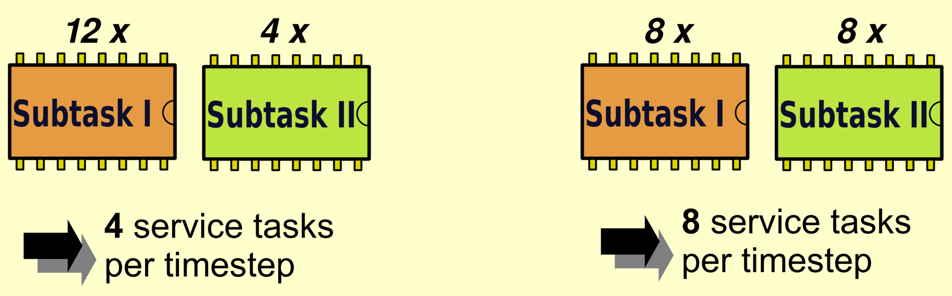


Components

- can switch their specialization to another type of subtask (reconfiguration)
- switching needs time
- switching strategies should use only local decisions
- overall aim: high system performance (number of executed service tasks / time)

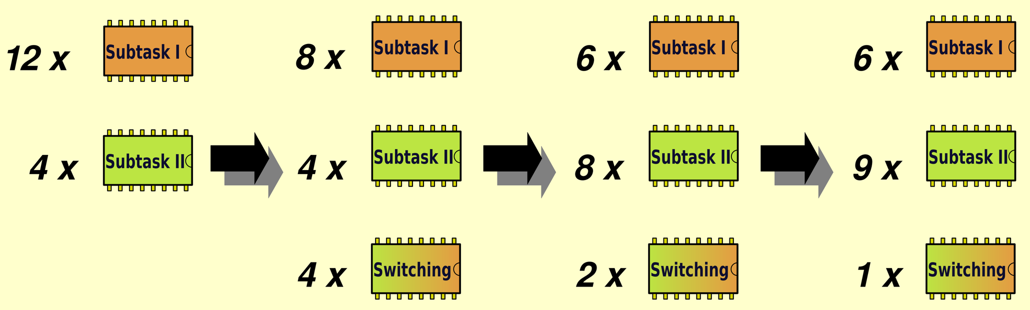
Simple Example: Two Subtasks

Equilibrium is important for Performance!



Switching Strategy: when waiting switch with probability p = 50%

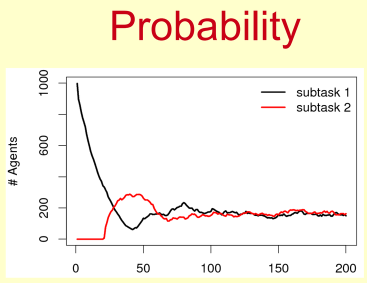
(Assumption: switch time $\tau = 2$ time steps)



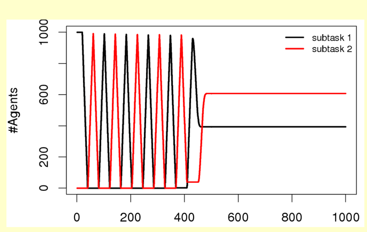
Investigation of Different Switching Strategies

- start with 1000 components specialized for subtask 1
- time for switch = 20

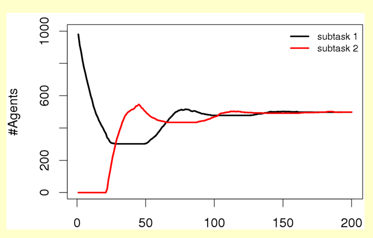
Same Fixed Switching Probability



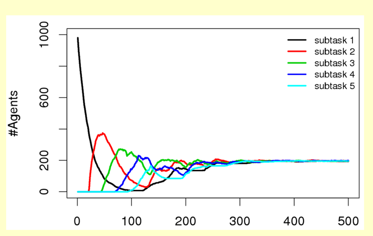
Same Fixed Thresholds for Maximum Waiting Time



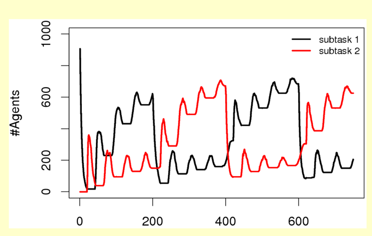
Thresholds for Maximum Waiting Time Exponentially Distributed



10 Subtasks



dynamic Scenario



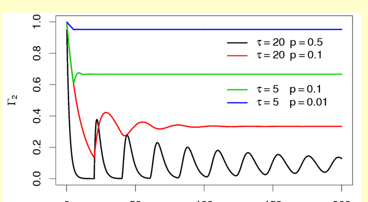
Theoretical Analysis

Stability Analysis

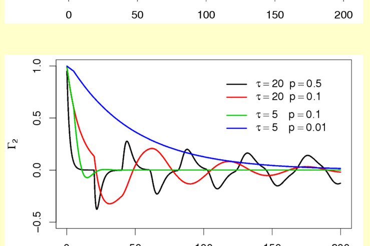
$$\frac{d\gamma_1}{dt} = -p\gamma_1(t) + p\gamma_2(t-\tau)$$

$$\frac{d\gamma_2}{dt} = -p\gamma_2(t) + p\gamma_1(t-\tau)$$

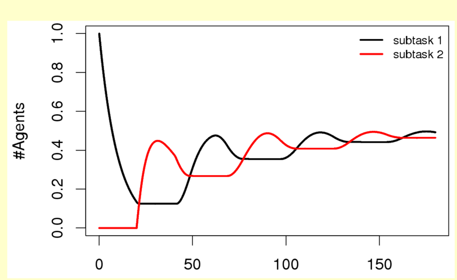
sum of active components for both type of subtasks is stable



difference of number of active components for different subtasks is asymptotically stable

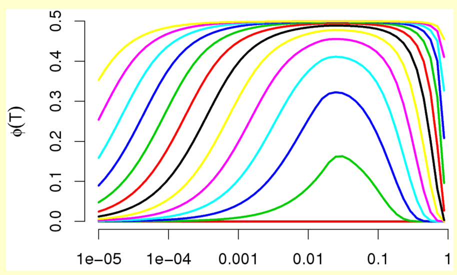


Numerical Investigation



system performance:

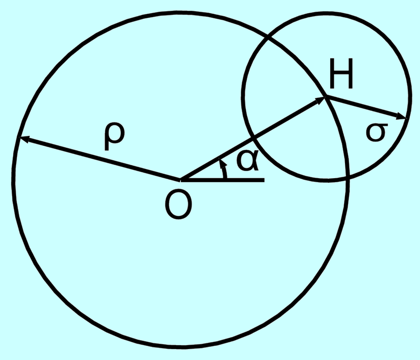
$$\Phi(T) = \frac{1}{T}\int_{t=0}^{T} min\left(\frac{\gamma_1(t)}{t_1}, \frac{\gamma_2(t)}{t_2}\right)$$

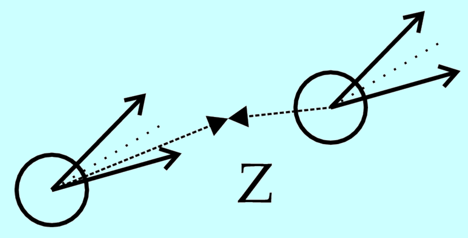


Congestion Control for Ant Like Moving Agents

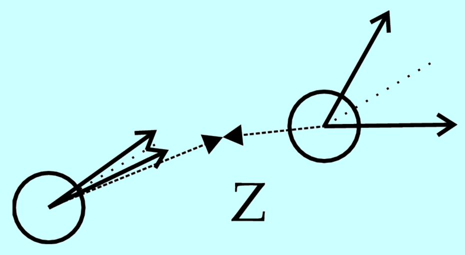
Agents

- head with sensing range
- simple ant inspired moving rules ("Chemotaxis")





full energy storage



nearly empty energy storage

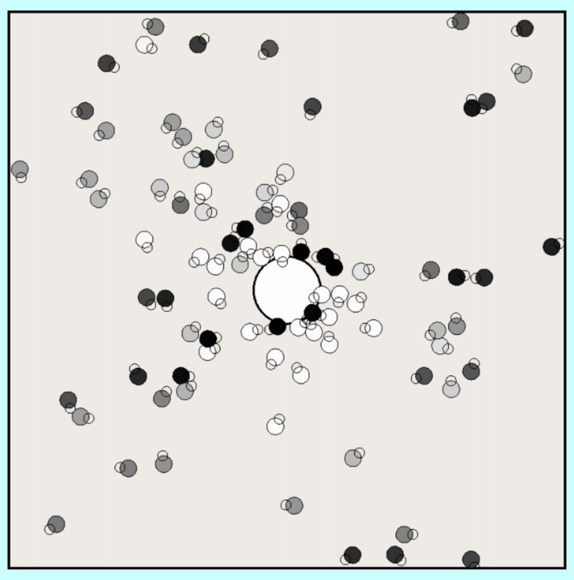
Energy Storage of Agent

- decreases when agent moves
- determines velocity
- determines turning angle
- can be refilled at service station Z

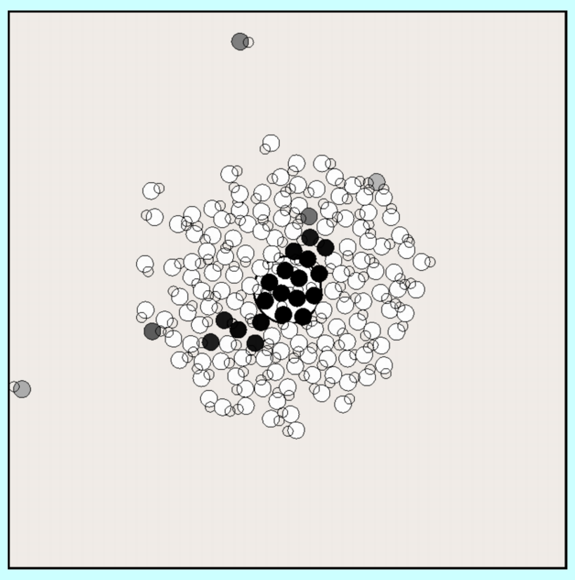
Emergent Congestion Effect

- dependent on the number of agents there can be a congestion at the service station

90 Agents



150 Agents

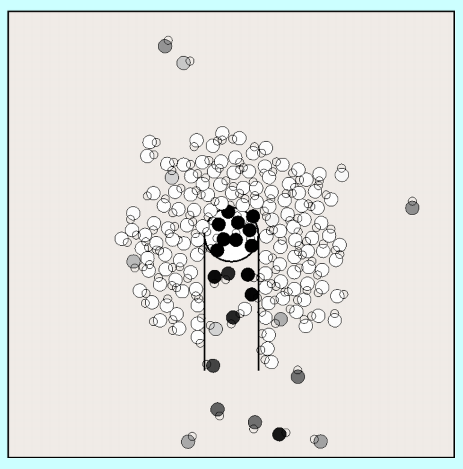


Investigation of Different Control Methods

Control Method C_P:

two neighboured walls

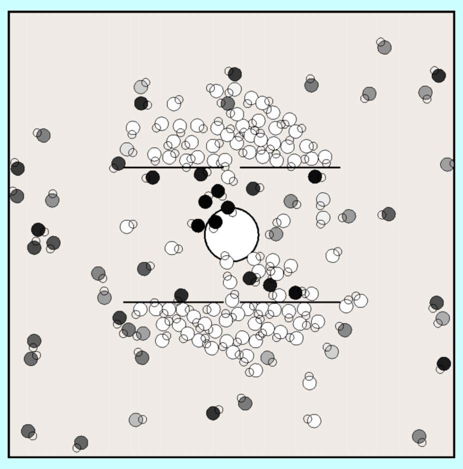
- walls "help" agents with full energy to leave the service station
- does not change agents behaviour



Control Method C_W:

walls with holes

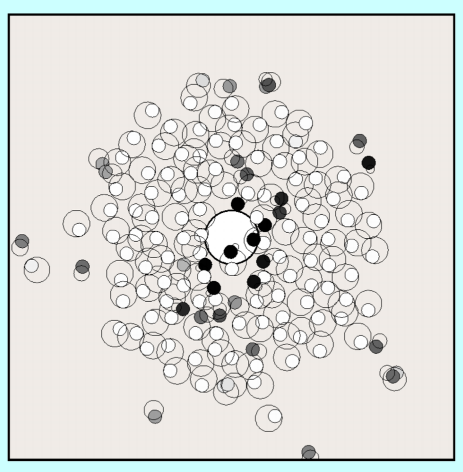
- walls control the traffic to the service station
- does not change agents behaviour

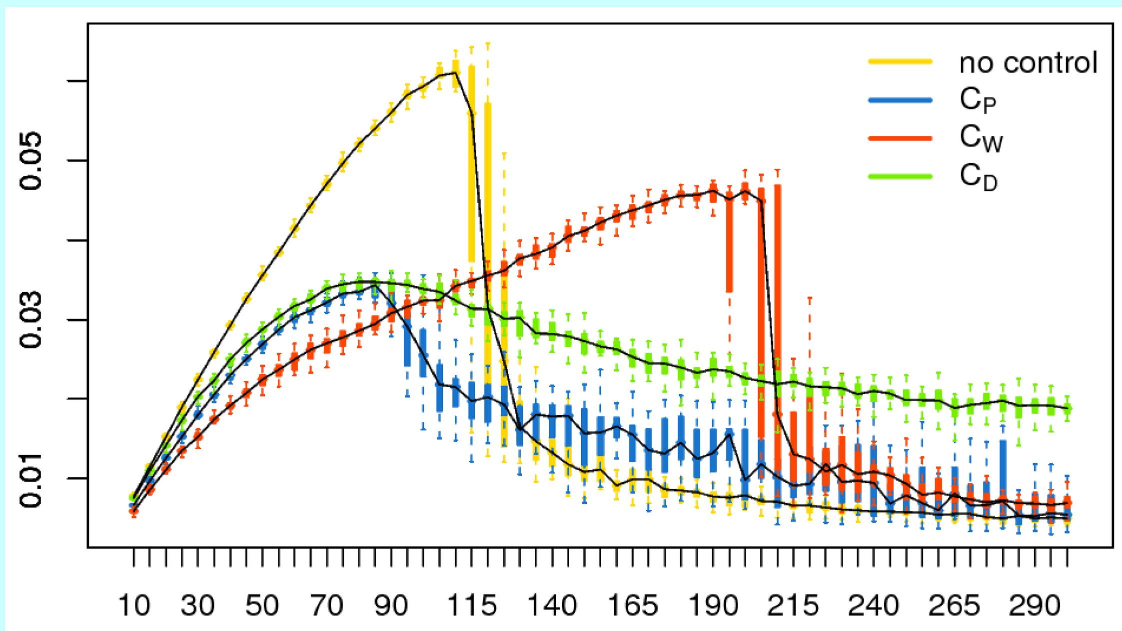


Control Method C_D:

sensing range of agents depends on stored energy

- larger sensing range of agents with empty store leads to space for refilled agents to leave
- changes agents behaviour





system performance for different number of agents

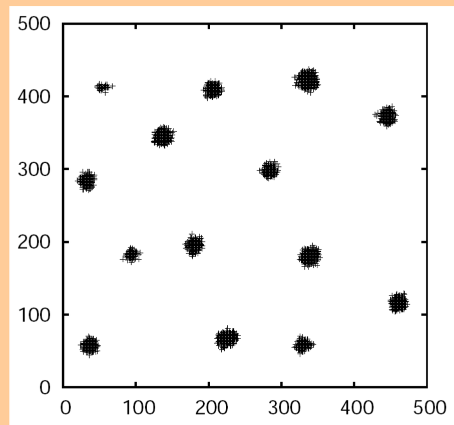
Swarm Controlled Emergence

Example: Designing an Anti-Clustering Ant System

Ant Clustering

- 2 dimensional grid with items
- agents move randomly
- probability for picking up and dropping items (f = fraction of occupied cells in neighbourhood)

$$p_p = \left(\frac{k_1}{k_1+f}\right)^2 \quad p_d = \left(\frac{f}{k_2+f}\right)^2$$



How to Control the Emerging Clustering ?

- control effect itself can be emergent
- using a swarm of anti-agents which have only slight different behaviour

Cluster Measures

Spatial Entropy

$$E_s = -\sum_{I \in s-patches} p_I \log p_I$$

- space divided into s-patches
- p_I is fraction of occupied cells in patch I

Hierarchical Social Entropy

$$S = -\int_0^{\infty} \sum_{i=0}^{M(h)} p_i \log p_i \, dh$$

- hierarchical clustering leads to a dendrogram
- M(h) number of groups at taxonomic level h
- p_i proportion of objects in the ith group

Ripley's K-Function

$$R(k) = \left(\frac{N}{n}\right)^2 \sum_{i=1}^{n} \sum_{j=1}^{n} I(d_{ij} < r)$$

- expected number neighbouring items within a given radius r

Summary Function

- F(r) – distribution function of the distance from any fixed cell to the nearest item (Empty Space Function)
- G(r) – cumulative distribution function of the distance from a fixed item to the nearest item (Nearest Neighbour Distance Distribution Function)
- estimation of Summary Function: $\hat{J}(r) = \frac{\hat{G}(r)}{\hat{F}(r)}$
- $\hat{J}(r) > 1$ indicates clustering
- $\hat{J}(r) < 1$ indicates order

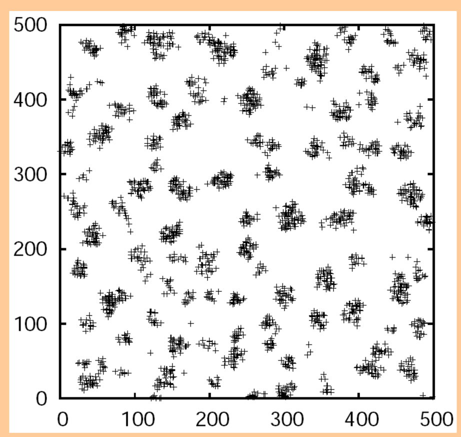
Anti Clustering Agents (AC-Agents)

Reverse AC-Agents

- opposite behaviour of standard agents
- probabilities exchanged

$$p_p = \left(\frac{f}{k_2+f}\right)^2 \quad p_d = \left(\frac{k_1}{k_1+f}\right)^2$$

Result: least effective ac-agents

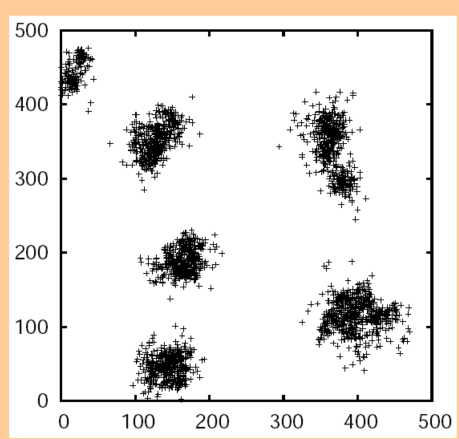


Random AC-Agents

- always pick up item
- drop item with fixed probability

$$p_p = 1 \quad p_d = \frac{1}{10}$$

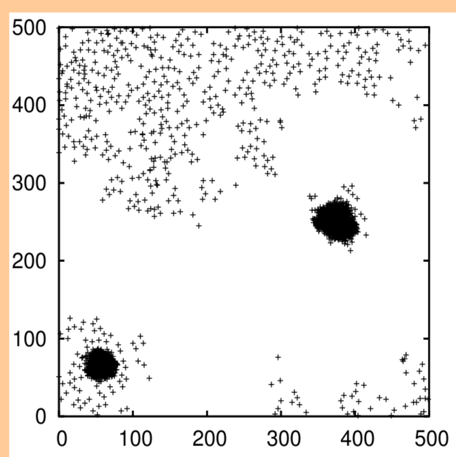
Result: can prevent clustering



Deterministic AC-Agents

- pick up item in case of any occupied cell in neighbourhood
- drop item in case of no occupied cell in the neighbourhood

$$p_p = \lceil f \rceil \quad p_d = \lfloor 1 - f \rfloor$$



Result: large number of this ac-agents can prevent clustering
Interesting effect: median number of this agent type can lead to better clustering