Observation and Control of Collaborative Systems (OCCS)

(Phase 2 of Quantitative Emergence (QE))

7th colloquium of the DFG SPP Organic Computing, Zurich, Switzerland
September 17th/18th, 2008

J. Branke, U. Richter, H. Schmeck
Karlsruhe Institute of Technology

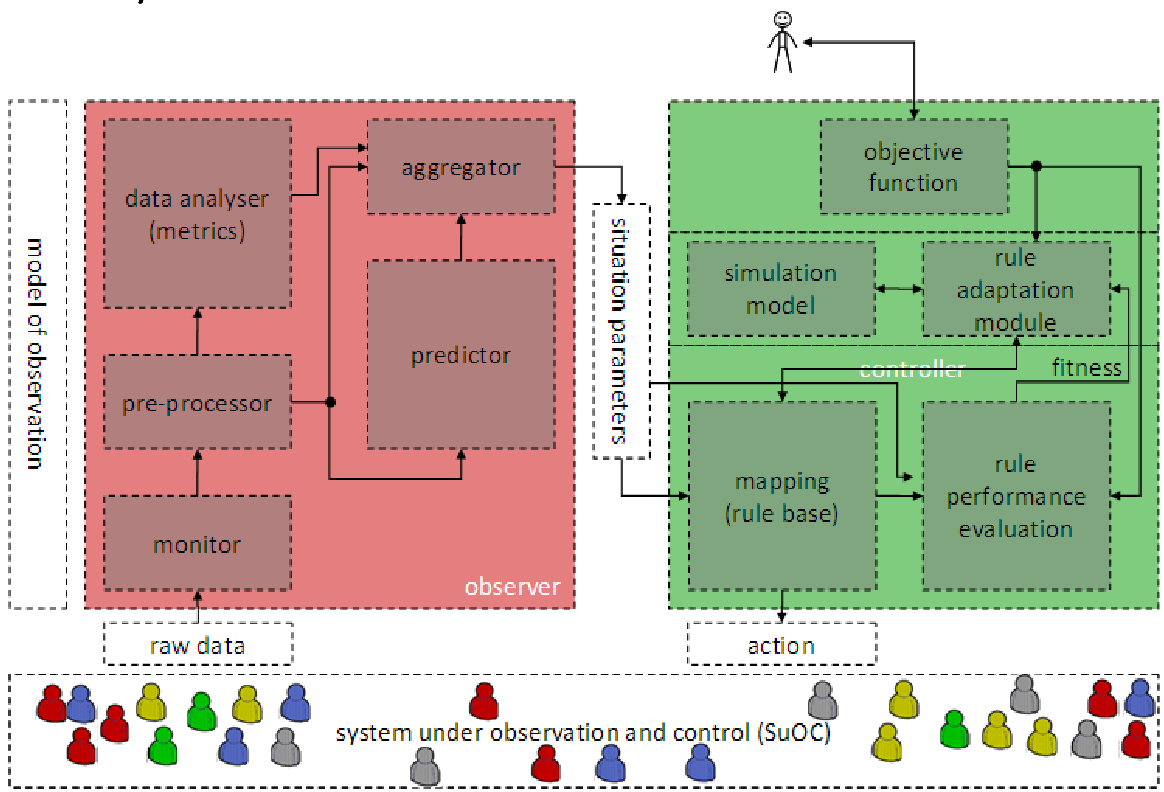
E. Cakar, J. Hähner, C. Müller-Schloer
Leibniz Universität Hannover

Outline

- Motivation
  - From phase I to phase II
- Deeper understanding of the O/C architecture
  - Systematic investigation of different distribution possibilities
  - Learning to control on-line with Learning Classifier Systems
- Conclusion and outlook
  - Remainder of phase II

Phase I

- Specification of the generic O/C architecture
- Goal: Establishing controlled self-organisation in technical systems
- Observer monitors and quantifies system states and dynamics.
- Controller influences the SuOC.



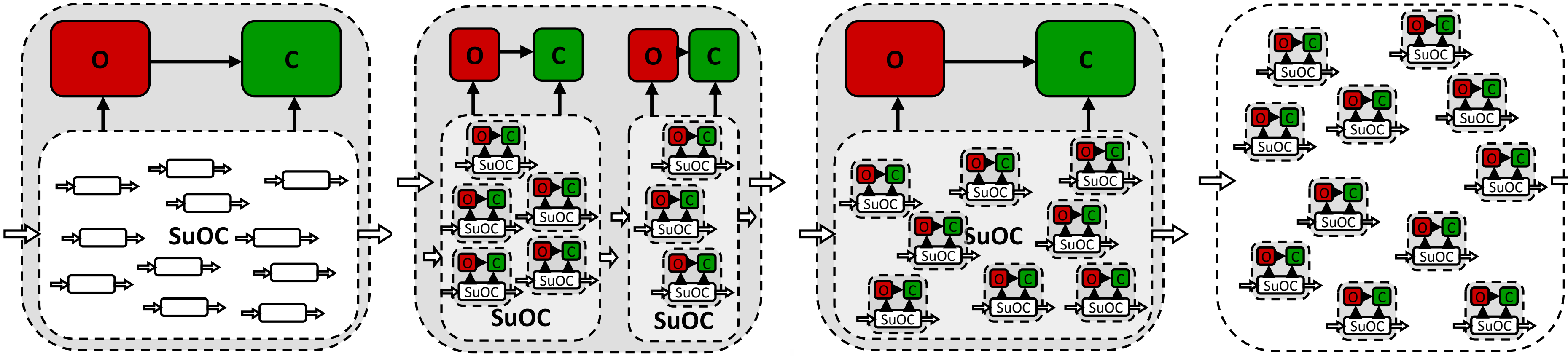
- Application of a **central** observer/controller architecture to a robot swarm

cf. Mnif, M., Richter, U., Branke, J., Schmeck, H., Müller-Schloer, C.: Measurement and control of self-organised behaviour in robots. In: Proceedings of the 20th International Conference on Architecture of Computing Systems (ARCS 2007).

Motivation of phase II

Focussing on the observation and control of collaborative OC systems

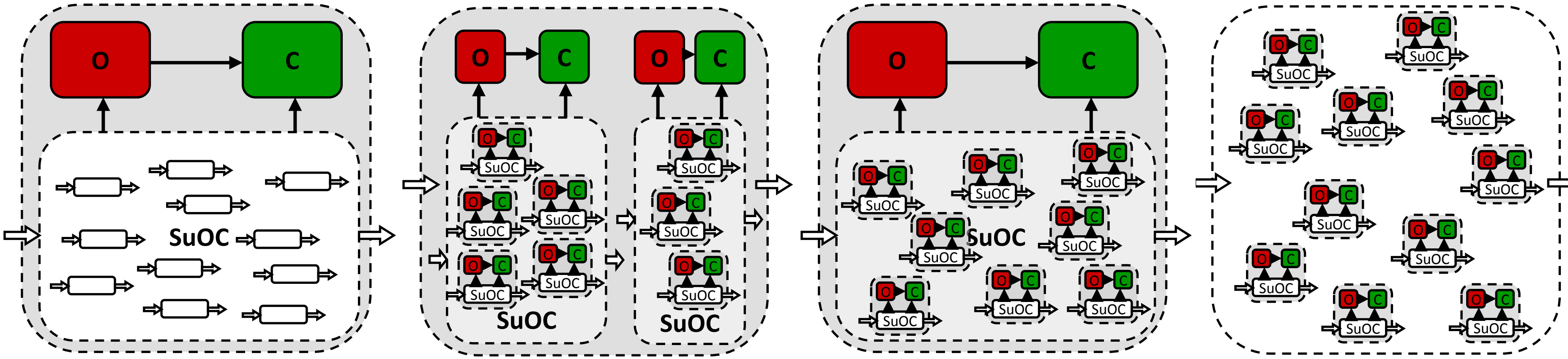
- Hannover
  - From centralised to distributed O/C architectures
  - Using distributed O/C architectures to create collaborative group behaviour
  - Systematic investigation of distribution patterns in a traffic scenario
- Karlsruhe
  - Investigation of on-line learning with Learning Classifier Systems (LCSs)
  - Parallel and hierarchical learning with eXtended Classifier Systems (XCSs)
  - Dealing with collective learning as part of the distributed controllers



Motivation of phase II

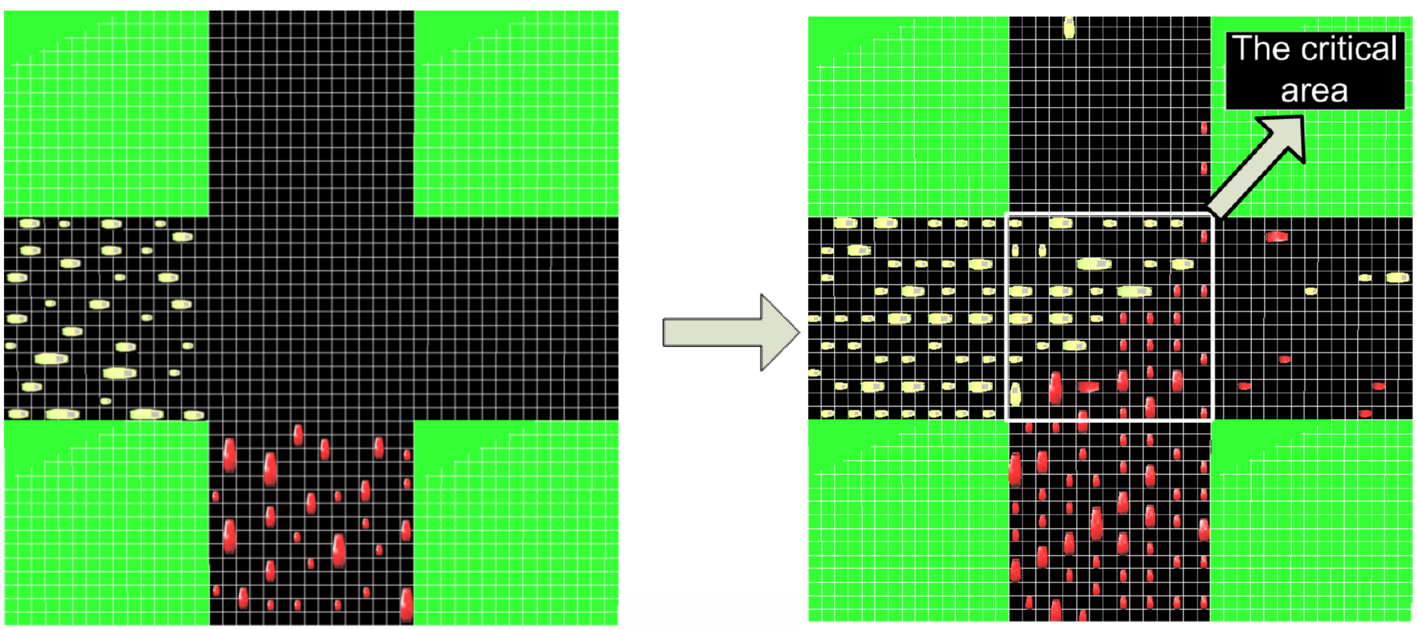
Focussing on the observation and control of collaborative OC systems

- Hannover
  - From centralised to distributed O/C architectures
  - Using distributed O/C architectures to create collaborative group behaviour
  - Systematic investigation of distribution patterns in a traffic scenario
- Karlsruhe
  - Investigation of on-line learning with Learning Classifier Systems (LCSs)
  - Parallel and hierarchical learning with eXtended Classifier Systems (XCSs)
  - Dealing with collective learning as part of the distributed controllers



Self-organising intersection

- Two traffic flows in west-east and south-north directions in an intersection without traffic lights
- Capabilities of a car without an O/C
  - Collision avoidance while moving
  - Local goal: Crossing the intersection as soon as possible
- No collaboration, but competition for limited resources



Traffic jam!

Self-organisation without control

Self-organising intersection

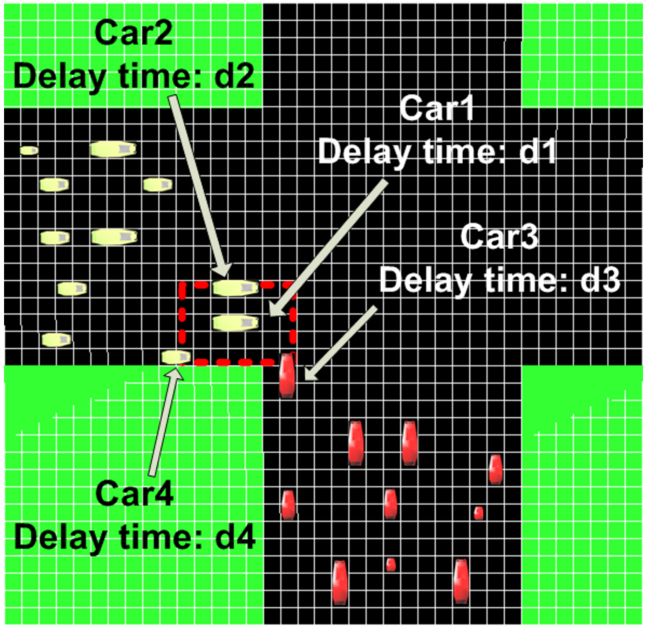
- Goal: Controlled self-organisation using O/C architectures
  - Collaboration instead of competition
  - Controlling the self-organisation process by preventing the competition and facilitating the collaboration using O/C architectures

- How to realise the collaboration?
  - Clustering problem in the critical area is a kind of scheduling problem.
  - Using a priority-based scheduling algorithm to realise the collaboration: A car (or a group of cars) with a higher delay time gets a higher priority.

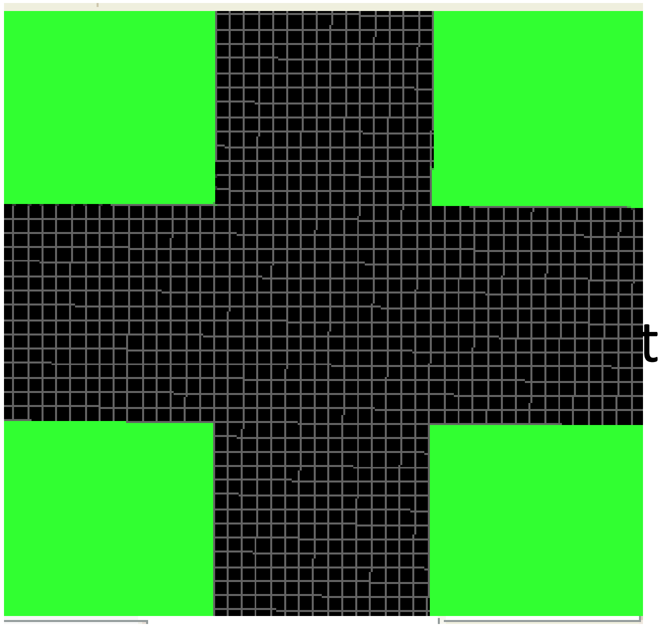
- Implementation of the algorithm on different distribution levels of the generic O/C architecture
  - A fully distributed O/C architecture
  - A centralised O/C architecture

A fully distributed O/C architecture

- The view of each car is limited to its direct neighborhood.
- Observer creates a list of situation parameters considering the direct neighborhood of the car.
- Controller makes its car collaborative.
- A sample situation:



Collaboration on the distributed level

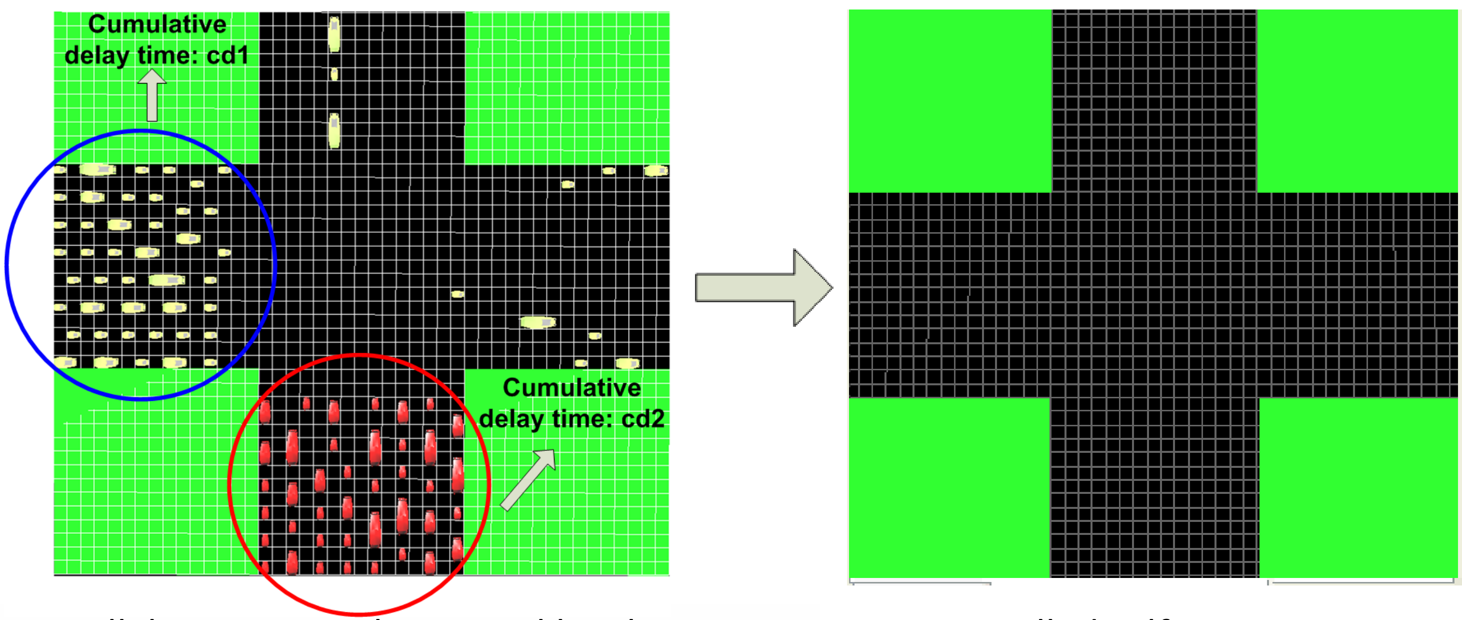


Controlled self-organisation

No traffic jam

A centralised O/C architecture

- One observer and one controller in the system with an unlimited view
- Assumption: Resource (cpu, memory, etc…) limitation on the central instance. → Behaviour of every single agent cannot be explicitly determined by the central controller.
- Assign priorities to **groups of cars** on a higher abstraction level.
- A sample situation:



No traffic jam

Collaboration on the central level

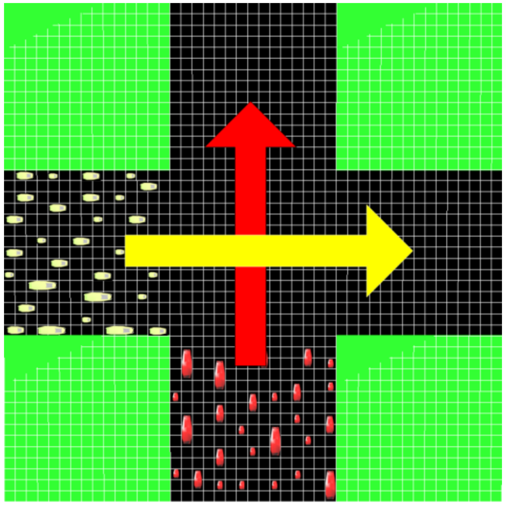
Controlled self-organisation

Centralised vs. fully distributed

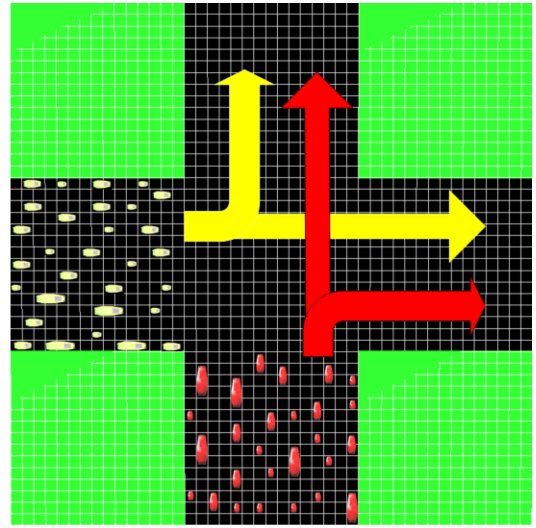
- Self-organising agents without an O/C architecture compete for limited resources.
  - Competition produces traffic jam.
- Self-organising agents with an O/C collaborate with each other.
  - Collaboration prevents traffic jam.

- **Question**: We want the agents to collaborate with each other, but which O/C architecture is better?
  - **Centralised vs. fully distributed.**
  - A comparison of both architectures is needed.
- Comparison criteria
  - System performance is measured with traffic-flow rate.
- 4 different test scenarios with different conflict levels to compare the architectures.

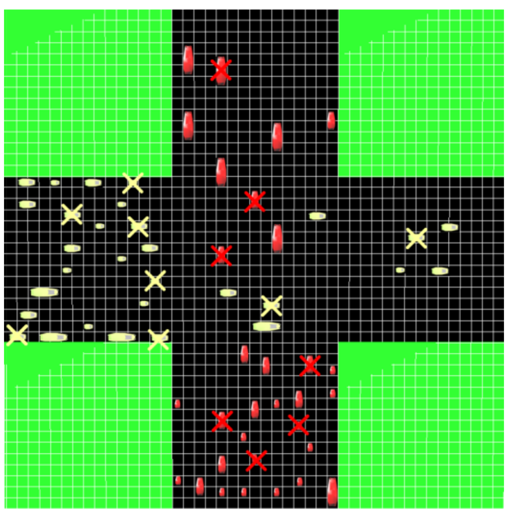
Test scenarios



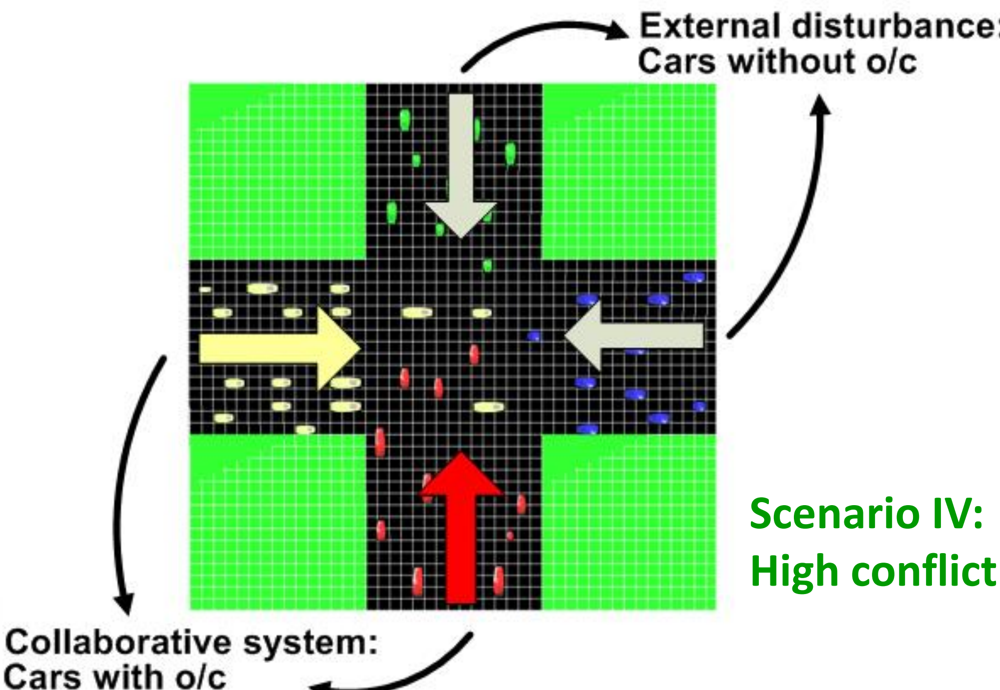
Scenario I: Low conflict level



Scenario II:
High conflict level



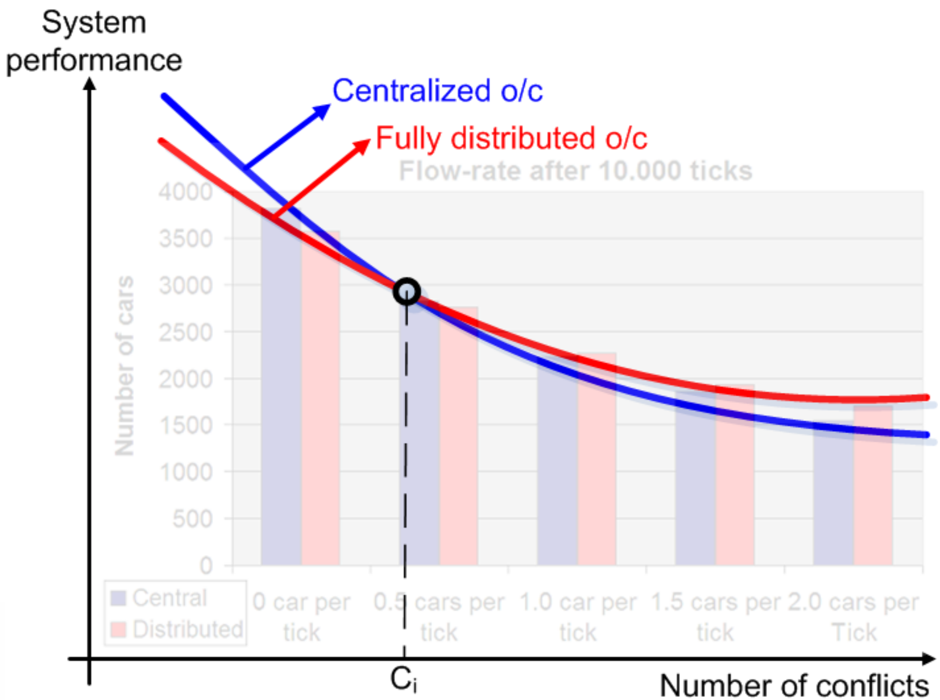
Scenario III: High conflict level



Scenario IV:
High conflict level

Intermediate results

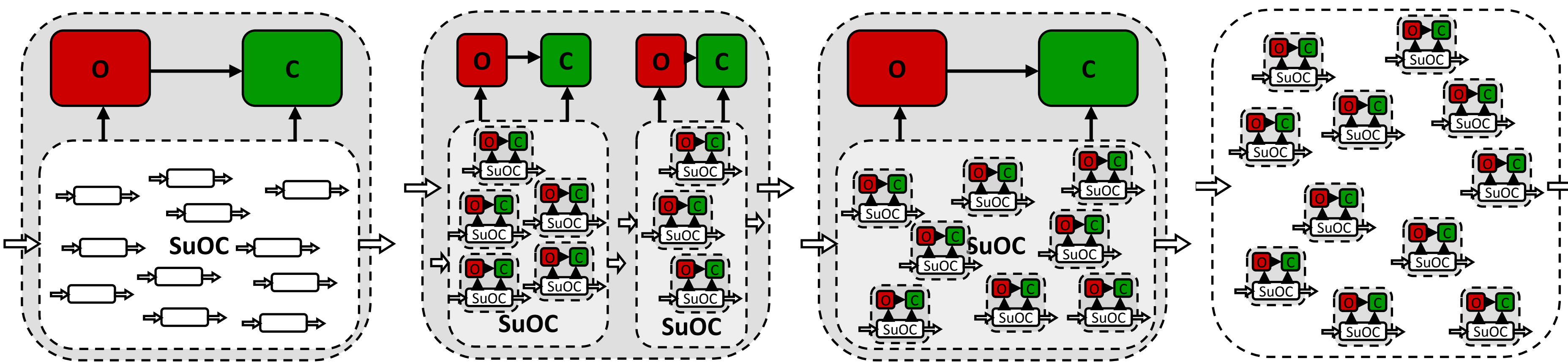
- Better system performance with the centralised O/C architecture in the low-conflict scenario (scenario I).
- Better system performance with the fully distributed O/C architecture in scenarios with high conflict level (scenarios II, III, and IV).
- The optimal collaboration strategy can neither be implemented on the central nor on the fully distributed level.
- Idea: An adaptive architecture that switches between the centralised and the fully distributed architecture depending on the conflict level.
- Outlook: Investigation of techniques to identify the switching criteria.



Motivation of phase II

Focussing on the observation and control of collaborative OC systems

- Hannover
  - From centralised to distributed O/C architectures
  - Using distributed O/C architectures to create collaborative group behaviour
  - Systematic investigation of distribution patterns in a traffic scenario
- Karlsruhe
  - Investigation of on-line learning with Learning Classifier Systems (LCSs)
  - Parallel and hierarchical learning with eXtended Classifier Systems (XCSs)
  - Dealing with collective learning as part of the distributed controllers

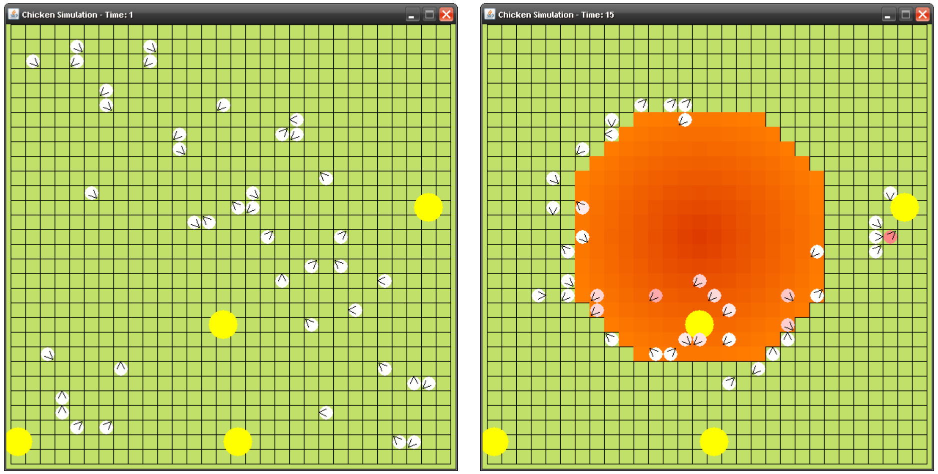


Learning scenario: Chicken simulation

- 40 agents (chickens)
- Playground with a dimension of 30 × 30 fields
- When a chicken is killed, a new chicken is generated and placed randomly in the cage.
- Observer:
  Situation parameters at every tick t

  $S_t = (e_x, e_y, e_h, (x_c, y_c))$
- Controller:
  Action (noise signal)

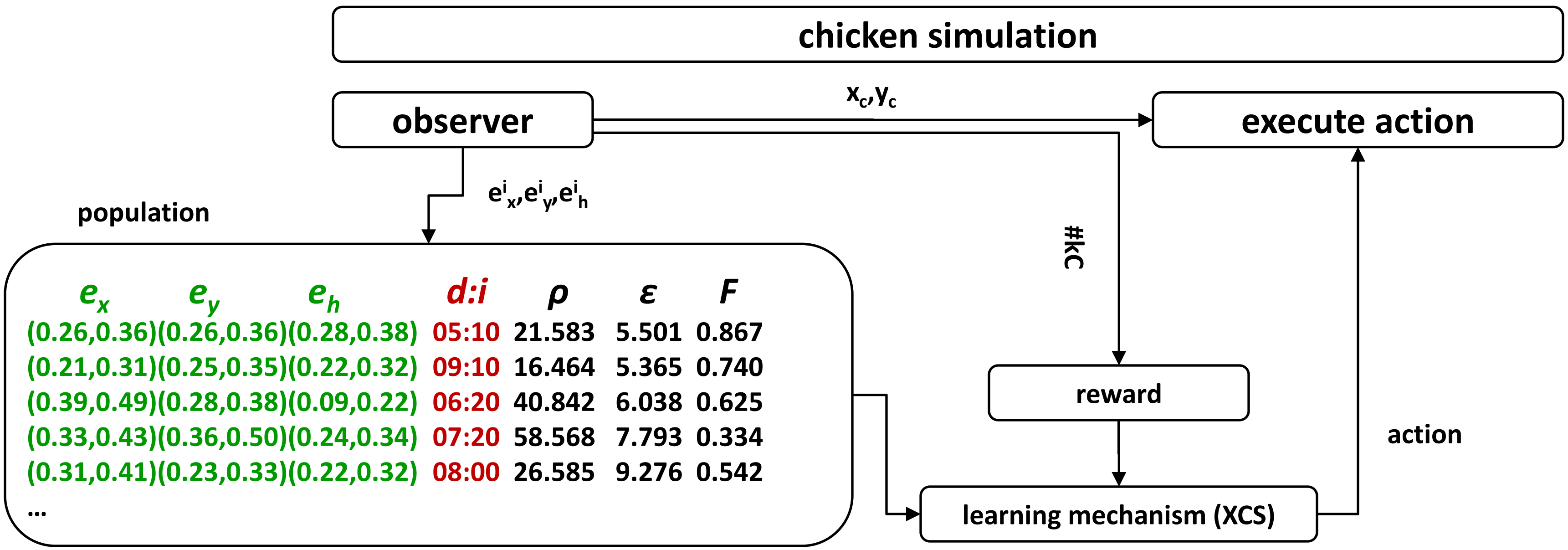
  $A = f(d, i, (x_c, y_c)) \approx f(d, i)$



- Learning: If and which noise signal should be applied?

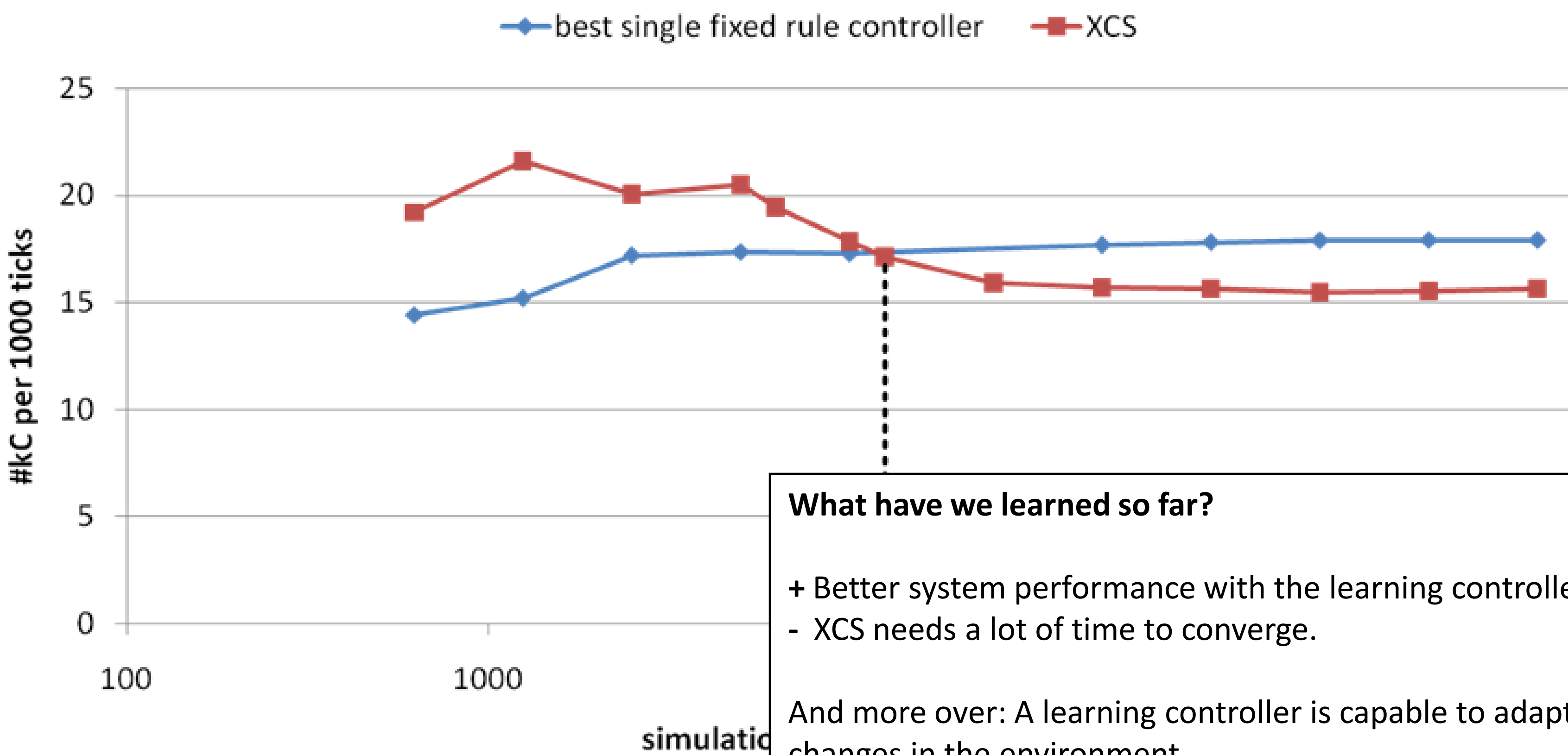
On-line learning with a learning classifier system (XCS)

- The idea of LCS fits well to the observer/controller architecture.
- Modified XCSJava1.0 reference implementation by Butz
- 20 seed values, a maximal population of 800 classifiers



e_x	e_y	e_h	d:i	ρ	ε	F
(0.26,0.36)	(0.26,0.36)	(0.28,0.38)	05:10	21.583	5.501	0.867
(0.21,0.31)	(0.25,0.35)	(0.22,0.32)	09:10	16.464	5.365	0.740
(0.39,0.49)	(0.28,0.38)	(0.09,0.22)	06:20	40.842	6.038	0.625
(0.33,0.43)	(0.36,0.50)	(0.24,0.34)	07:20	58.568	7.793	0.334
(0.31,0.41)	(0.23,0.33)	(0.22,0.32)	08:00	26.585	9.276	0.542
…						

Learning over time LCS vs. the best single fixed rule controller



What have we learned so far?

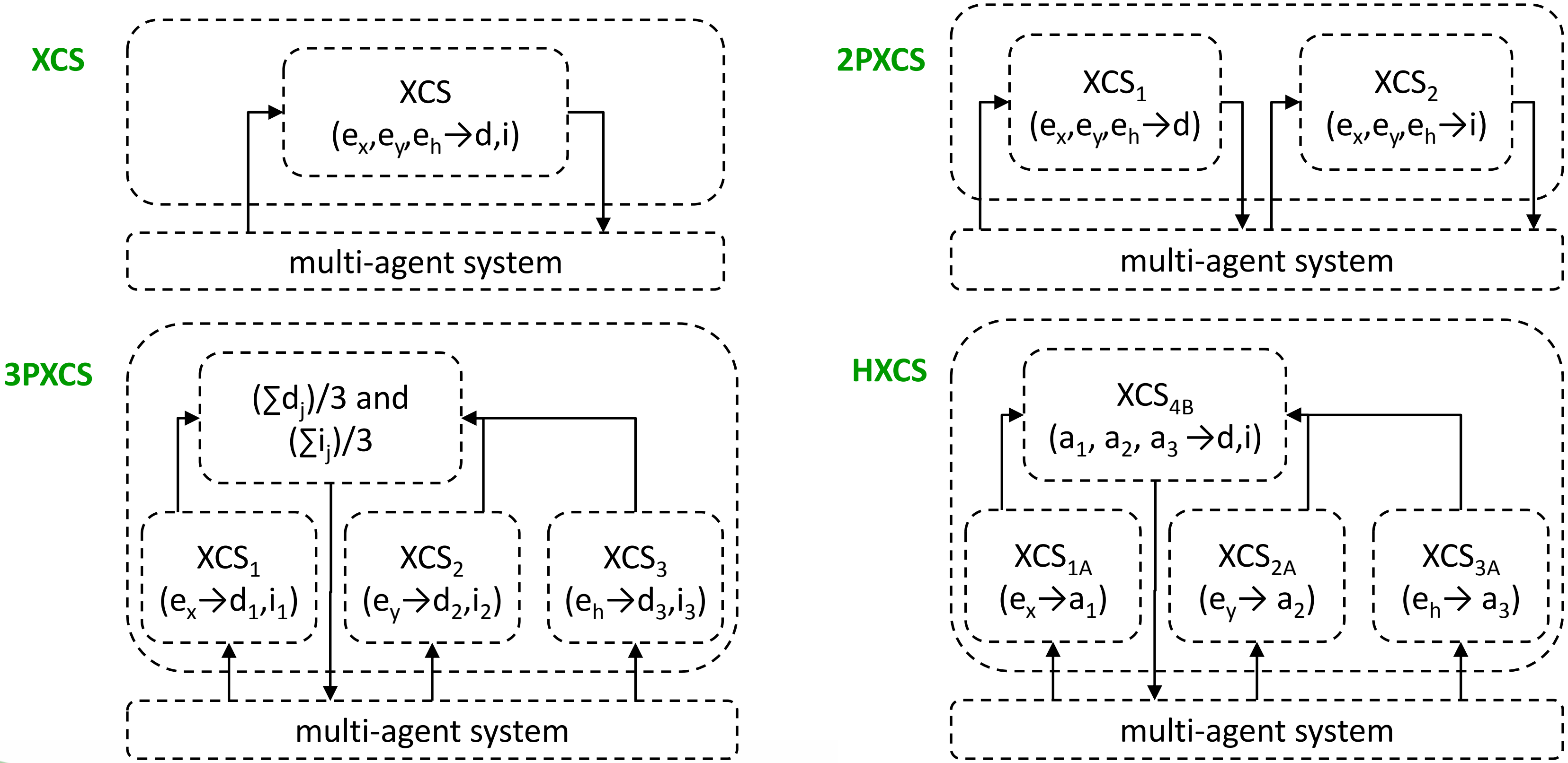
\+ Better system performance with the learning controller.
\- XCS needs a lot of time to converge.

And more over: A learning controller is capable to adapt to changes in the environment.
Question: How can we increase the learning rate of XCS?

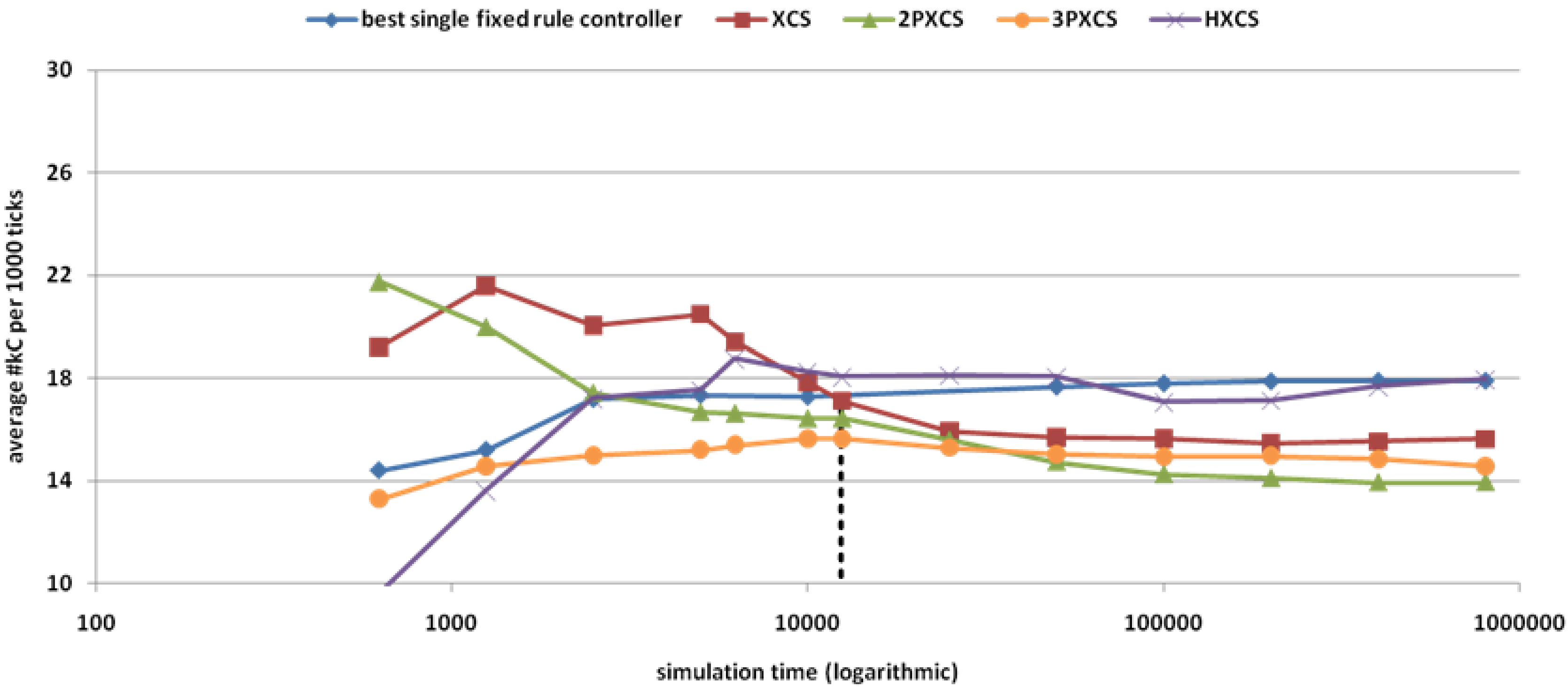
ordinate = (average #kC / simulated ticks) * 1 000

Parallel and hierarchical architectures

- Idea: Improve the global XCS performance by splitting the options of condition-action-mappings into smaller sub-mappings and by solving/combining them with parallel collaborative LCSs.



Intermediate results



- XCS needs (a lot of) time to converge.
- Increasing objective spaces force problems.
- Parallel and hierarchical LCS implementations seem to be promising.

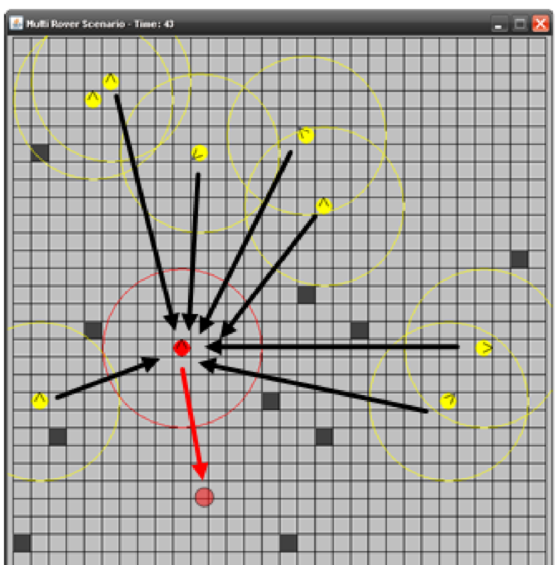
Conclusion & Outlook

Remainder of phase II

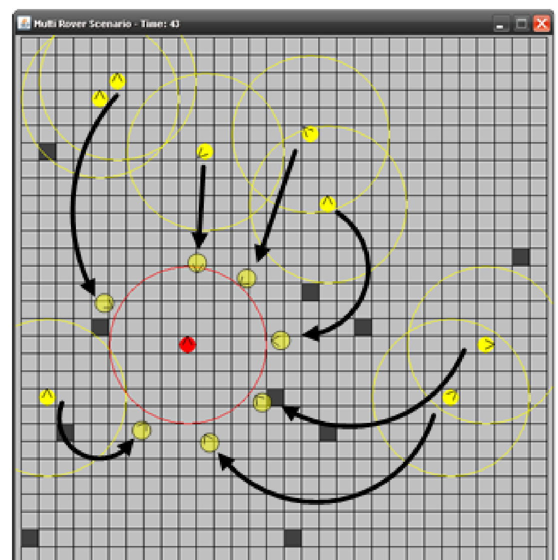
- Distribution aspects of the O/C architecture
- Collective learning aspects

→ Collaboration patterns

- Multi rover scenario
  - 2D grid world with obstacles
  - A number of rovers has to find and to observe one ore more targets .



Without collaboration



With collaboration

Recent publications (1/2)

2008

- Branke, J. and Schmeck, H. 2008. **Evolutionary design of emergent behavior.** In Organic Computing, Würtz, R. P., Eds. Springer, 123–140.
- Cakar, E., Hähner, J., and Müller-Schloer, C. 2008. **Investigation of generic observer/controller architectures in a traffic scenario.** Accepted for publication in INFORMATIK 2008 – Beherrschbare Systeme – dank Informatik.
- Cakar, E., Hähner, J., and Müller-Schloer, C. 2008. **Creating collaboration patterns in multi-agent systems with generic observer/controller architectures.** Accepted for publication in Proceedings of the 2nd International ACM Conference on Autonomic Computing and Communication Systems (Autonomics 2008).
- Müller-Schloer, C. and Sick, B. 2008. **Controlled emergence and self-organisation.** In Organic Computing, Würtz, R. P., Eds. Springer, 81–104.
- Ribock, O., Richter, U., and Schmeck, H. 2008. **Using Organic Computing to control bunching effects.** In Proceedings of the 21th International Conference on Architecture of Computing Systems (ARCS 2008), U. Brinkschulte, T. Ungerer, C. Hochberger, and R. G. Spallek, Eds. LNCS, vol. 4934, Springer, 232–244.
- Richter, U. and Mnif, M. 2008. **Learning to control the emergent behaviour of a multi-agent system.** In Proceedings of the 2008 Workshop on Adaptive Learning Agents and Multi-Agent Systems at AAMAS 2008 (ALAMAS+ALAg 2008), F. Klügl, K. Tuyls, and S. Sen, Eds. 33 – 40.
- Richter, U., Prothmann, H., and Schmeck, H. 2008. **Improving XCS performance by distribution.** Accepted for publication in Proceedings of the 7th International Conference on Simulated Evolution And Learning (SEAL 2008).
- Schmeck, H. and Müller-Schloer, C. **A characterisation of key properties of environment-mediated multi-agent systems.** In Engineering Environment-Mediated Multi-Agent Systems. Danny Weyns, Sven Brueckner, Yves Demazeau (Eds.), LNCS, 2008.

2007

- Cakar, E., Mnif, M., Müller-Schloer, C., Richter, U., and Schmeck, H. 2007. **Towards a quantitative notion of self-organisation.** In Proceedings of the 2007 IEEE Congress on Evolutionary Computation (CEC 2007), 4222–4229.

Recent publications (2/2)

- Mnif, M., Richter, U., Branke, J., Schmeck, H., and Müller-Schloer, C. 2007. **Measurement and control of self-organised behaviour in robot swarms.** In Proceedings of the 20th International Conference on Architecture of Computing Systems (ARCS 2007), P. Lukowicz, L. Thiele, and G. Tröster, Eds. LNCS, vol. 4415. Springer, 209–223.

2006

- Branke, J., Mnif, M., Müller-Schloer, C., Prothmann, H., Richter, U., Rochner, F., and Schmeck, H. 2006. **Organic Computing – Addressing complexity by controlled self-organization.** In Post-Conference Proceedings of the 2nd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2006), T. Margaria, A. Philippou, and B. Steffen, Eds. Paphos, Cyprus, 185–191.
- Mnif, M. and Müller-Schloer, C. 2006. **Quantitative emergence.** In Proceedings of the 2006 IEEE Mountain Workshop on Adaptive and Learning Systems (IEEE SMCals 2006). 78–84.
- Müller-Schloer, C. and Sick, B. 2006. **Emergence in Organic Computing systems: Discussion of a controversial concept.** In Proceedings of the 3rd International Conference on Autonomic and Trusted Computing (ATC 2006), L. T. Yang, H. Jin, J. Ma, and T. Ungerer, Eds. LNCS, vol. 4158. Springer, 1–16.
- Richter, U., Mnif, M., Branke, J., Müller-Schloer, C., and Schmeck, H. 2006. **Towards a generic observer/controller architecture for Organic Computing.** In INFORMATIK 2006 – Informatik für Menschen!, C. Hochberger and R. Liskowsky, Eds. GI-Edition – Lecture Notes in Informatics (LNI), vol. P-93. Köllen Verlag, 112–119.

2005

- Müller-Schloer, C. 2005. **Organic Computing – Systemforschung zwischen Technik Naturwissenschaften**. it Special Issue on Organic Computing 47, 179–181.
- Schmeck, H. 2005a. **Organic Computing.** Künstliche Intelligenz 3, 68–69.
- Schmeck, H. 2005b. **Organic Computing – A new vision for distributed embedded systems.** In Proceedings of the 8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2005). IEEE Computer Society, 201–203.

2004

- Müller-Schloer, C. 2004. **Organic Computing: On the feasibility of controlled emergence.** In Proceedings of the 2nd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS 2004), 2–5.