Proceedings of the 2007 IEEE Swarm Intelligence Symposium (SIS 2007)

Swarm Controlled Emergence - Designing an Anti-Clustering Ant System

Daniel Merkle, Martin Middendorf, Alexander Scheidler
Parallel Computing and Complex Systems Group
University of Leipzig, Johannisgasse 26, 04103 Leipzig, Germany
Email: {merkle, middendorf, scheidler}@informatik.uni-leipzig.de

Abstract— **A new approach to prevent negative emergent behaviors of adaptive or organic computing systems is presented. One characteristic of such computing systems is the use self-organisation principles from nature and components that make decentralized decisions. To control such systems is a difficult task. In this paper we propose to control by introducing a swarm of so called anti-components to the system that can prevent the negative emergence. As an example serves a model that is inspired by the emergent behavior of ants to cluster different items. This model system has been used for several applications in computer science already. Different types of anti-components (or anti-agents) that can prevent a clustering behavior are designed for this system. Several cluster validity measures are used to investigate the clustering behavior of a system that contains standard clustering agents together with anti-clustering agents. It is shown that such systems can show a complex behavior over time where a phase of item distributions with increasing order is followed by distributions with increasing degree of clustering. It is also shown that a medium number of certain anti-clustering agents (which in a larger number completely prevent any clustering) may even help the system to perform a good clustering faster.**

I. Introduction

Emergent behavior is a well known phenomenon that occurs in many biological systems. Social insects are an example where several collective activities like foraging or nest building can be called emergent. Emergent phenomena in natural systems receive an increasing interest of computer scientists in recent years because their underlying principles have a potential value for the design of bio-inspired algorithms or bio-inspired computing systems. One example is the trail laying behavior of ants that leads to short paths between an ants nest and the food sources. This behavior has inspired the Ant Colony Optmization metaheuristic that is used to solve combinatorial optimization problems. Another example is the behavior of ants to cluster larvae or dead corpses which has inspired the design of different clustering algorithms.

Organic computing (OC) (see, e.g., [1], [21], [26]) for example is a new field of computer science with the aim to construct computing system that have so called self-x properties (where "x" stands, e.g., for "healing", "managing", "organizing", "optimizing") and which uses principles of self-organization from natural systems. One aspect that is considered important for such systems is their emergent behavior (see, e.g., [25]). Emergent behavior can be a positive or negative property of such systems and there exist efforts to develop quantitative measure for emergence ([20]). But so far researchers have considered mostly the positive aspects of emergent behavior. They have applied the principles of emergent behavior of natural systems to increase the capabilities of technical systems or to design algorithms with improved behavior.

Recently, some concerns came up that self-organized computing systems which consist of many autonomous components might show an emergent behavior that is neither wanted nor has it been intended or foreseen to occur when the systems were designed. In accordance with other authors we use the term negative emergence for such unwanted emergence behavior (see also [22] for a discussion of emergence). One important research question is how negative emergent behavior of OC systems can be prevented.

The only approach to prevent negative emergence in Organic Computing that has been proposed by several researchers is to equip OC-Systems with a so called observer controller subsystem ([23], [28]) where a set of observers collects information about the system and based on this information the controllers send control information to the components to influence their behavior. Not much concrete work has been done so far in this area. It has been proposed that the observer controller subsystem might consist of a hierarchy of observer controller elements where each controller computes control information (e.g. on the basis of a rule based process) that it sends to those components of the OC system which it controls. From a local point of view the observer controller approach relies mainly on classical control mechanisms. But from a global point of view an observer controller approach differs from many classical control approaches since it tries to apply its control functions as distributed as possible.

It is not questioned in this paper that the observer controller approach could be suitable for many OC systems. But we point out here that there is a principle disadvantage of this approach, namely that it relies fundamentally on (classical) controllers that send control messages to the components. Clearly, in a hierarchy of observer controller elements it might be possible to restrict the actual control influence locally so that only a small subset of the components receives certain control statements. Nevertheless, the whole control function in an observer controller architecture is realized via direct control of the components. Hence, control is obtained only via direct restriction of the autonomy of the single components of an OC-system. Since this works against some central principles of OC

1-4244-0708-7/07/$20.00 ©2007 IEEE

like self-organisation and self-autonomy, it is argued here that it is important to develop alternative approaches for controlling OC systems in order to prevent negative emergence. It is unlikely that there exists a single approach that works perfect for all applications. It will rather be necessary to have a bundle of control approaches that can be used for different OC systems.

In this paper we develop an alternative approach to prevent negative emergence in OC systems. The general idea is to design so called anti-emergence components (anti-components) which can prevent the occurrence of certain negative emergence effects when they are added to the OC system. One of the challenges is to design anti-components that behave not too different from the normal components of the OC system. Ideally, then the anti-components can still do normal work in the OC system (but eventually less efficient than the normal components - otherwise all components could become anti-components). We call our approach to prevent negative emergence *swarm controlled emergence*.

For our approach there exist two possibilities to determine the anti-components: i) anti-components can be permanently added to the OC systems so that negative emergence can not occur or, ii) the behavior of some components can be changed from normal mode to anti mode (i.e., a normal component becomes an anti-component) when negative emergence occurs. The switching can be realized either by one or more observers that send some components a message to switch to anti mode (or back to normal mode) or self-organized so that each components decides by itself whether to switch from normal mode to anti mode or vice versa.

Our approach to control negative emergence in OC systems differs fundamentally with respect to the following aspects from the observer controller approach:

i) It does not restrict the autonomy of the components (neither of the normal components nor of the anti-components) or at most requires them two switch between two modes.
ii) It does not assume that there exists a special type of observer controller element nor is it necessary to have a corresponding hierarchical communication structure for delivering control information.
iii) It does not need control information at all (this is case when the switch between normal mode and anti mode is self-organized) or needs only one type of control message that contains one bit of control information (which determines whether to switch from normal mode to anti mode or vice versa).

The aim of this paper is to explain the principle ideas of our approach, give a proof of concept for a test system, and start investigations on special properties of the new approach. Even when potential applications in OC are an important motivation for this study we consider the approach to be of general interest for the study of emergent systems. Therefore, it is not the intention to apply our approach directly to one of various architectures that are studied in the field of OC.

As a test system for this study we pick one of the famous examples of emergent behavior of social insects which has several applications in computer science — the clustering behavior of ants (see [10], [11], [12], [13], [14], [15], [16], [17]). One advantage of this system is that the agents have simple rules for their behavior and the system is well investigated. Another advantage is that ant clustering has not only been applied to solve combinatorial problems (e.g., clustering and sorting) but also to study emergence in robotics (e.g., [7]).

In the next Section we give a short overview on ant clustering. Section III describes anti-clustering agents. Cluster validity measures are presented in Section IV. Results are presented in Section V and conclusions are given in Section VI.

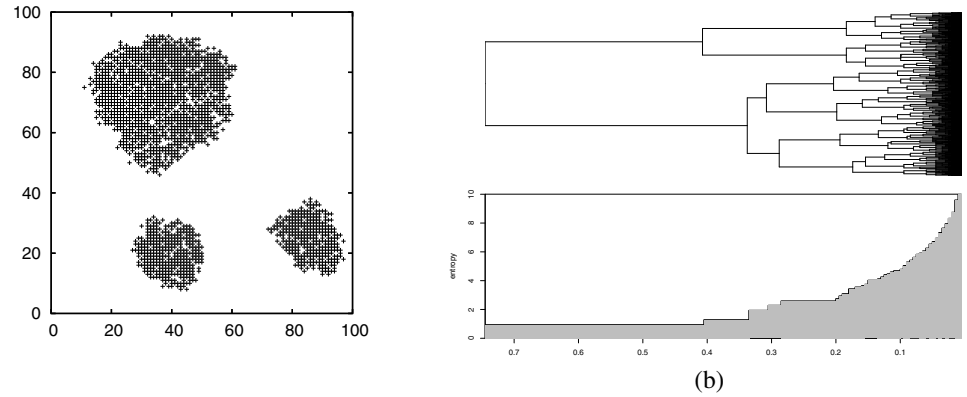
II. Ant Clustering

For the ant species Lepthotorax unifasciatus it is well studied that the brood is organized in concentric rings according to their size around the nest center during the brood tending process [8]. Another emergent phenomenon of ants are their cemeteries (see [5], [7], [29]). When corpses of dead ants are randomly distributed in a two-dimensional area they are organized into clusters by the ants within a few hours. Both phenomena have been addressed by simple multi-agent models. In a model that was proposed in [7] several items are distributed in a two-dimensional array of cells (at most one item per cell). Each agent walks randomly within the cell array, picks up an item that it finds with a certain probability, carries it around and let it fall with a certain probability. Formally, the probability p_{p_p} for an unladen agent to pick up an item is $p_p = \left(\frac{k_1}{k_1+f}\right)^2$, where f is the fraction of cells in the neighborhood of the agent that are occupied with items and k_1 is a threshold value. Analogously, the probability that a laden agent drops an item if it is on a cell that is not occupied by an item is given by $p_d = \left(\frac{f}{k_2+f}\right)^2$, with k_2 being a threshold value. Several methods for calculating the value f have been proposed. One method is to count how many items have been encountered by the agent during the last time steps and define f as the fraction of time steps where the agent moved across cells that are occupied by an item. Another way to calculate f is to calculate the fraction of cells in the von Neumann neighborhood of the agent that is occupied with item.

Although this simple model is not able to reproduce the pattern of concentric rings that occurs in the brood sorting process of Lepthotorax, the model fits the ants clustering behavior during the organization of cemeteries very well. For more elaborated models for explaining brood sorting patterns see, e.g., [27] or [19].

III. Anti-Clustering

In this paper we consider the emergent clustering effect in the ant model that was described in the last section as an unwanted negative emergent effect. Clearly, the clustering effect has a biological relevance and it can be used positively for various applications in computer science. But it is also possible to consider it to be unwanted (even though this has



(a) (b)

Fig. 1. Hierarchical social entropy; depicted are an exemplary clustering situation on a 100×100 field (a), the resulting dendrogramm (b, upper), and the value of the social entropy at different taxonomic levels (b, lower)

never been done so far in the literature). In order to prevent the clustering effect we assume that there does not exist any control system that can directly influence the agents behavior. Instead, we want to add agents to the system that behave similar to the standard agents but can prevent (or reduce) the clustering effect. These agents are called here anti-clustering agents, or $\mathcal{AC}$-agents. In the following we introduce different types of anti-clustering agents.

Reverse $\mathcal{AC}$-agents. An intuitive idea to prevent clustering is to introduce agents that have a behavior which is opposite to the behavior of the standard agents. The following two intuitive methods are considered for $\mathcal{AC}$-agents: i) The two probabilities that an agent picks up an item or drops an item in a certain situation are exchanged. These agents are called reverse agents (or type 1 agents). ii) Let λ be the fraction of cells that are occupied by an item in the neighborhood of an agent. If $p(\lambda)$ is the probability that a standard agent picks up (respectively drops) an item, then an $\mathcal{AC}$-agent of type 2 picks up (respectively drops) an item with probability $p(1-\lambda)$.

Random $\mathcal{AC}$-agents. Introducing sufficient randomness in the clustering process in the sense that items are placed on random cells can obviously hinder a strong clustering. The second class of $\mathcal{AC}$-agents pick up an item always when they enter a cell that is occupied by an item. If such an agent carries an item it drops it with a fixed probability 0.1.

Deterministic $\mathcal{AC}$-agents. The third class of $\mathcal{AC}$-agents use a deterministic strategy. An agent always picks up the item if it enters a cell that is occupied by an item and always drops the item if no item is in the neighborhood of the current cell.

IV. Cluster Validity and Clustering Measures

One central aspect for our study is to measure how the strength of an emerging clustering changes. Therefore, several measures for the degree of clustering that have been proposed in the literature are described in this section.

Spatial Entropy. In [9] and [4] it was suggested to use spatial entropy to track the dynamics of clustering. The spatial entropy is a measure to classify spatial distributions of items according to their cluster validity on different spatial scales. Therefore, the (two-dimensional) cell array A is partitioned into so-called s-patches, i.e., subarrays of size $s \times s$. Let p_I be the fraction of cells in an s-patch I that are occupied by an item. Then the spatial entropy E_s at scale s is defined as

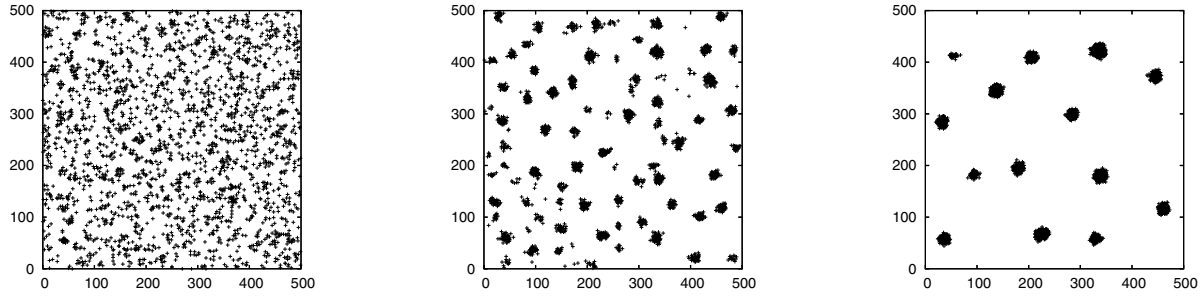
$$E_s = - \sum_{I \in \{s-\text{patches}\}} p_I \log p_I$$

Ripley's K-function. Ripley's K-function was defined in [24] in the context of spatial point processes. Let λ be the intensity of the point process (i.e., the expected number of points per unit area). Ripley's K-function (also called reduced second moment function) is defined such that $\lambda \cdot K(r)$ equals the expected number of additional points within a distance r of a randomly chosen point of the point process. For a given distribution of items in a two-dimensional cell array of size A the value $K(r)$ can be estimated. This is done for a given distribution of n items in an $N \times N$ cell array as follows

$$\hat{K}(r) = (\frac{N}{n})^2 \sum_{i=1}^{n} \sum_{j=1}^{n} I(d_{ij} < r)$$

where $I()$ is the indicator function, i.e., $I(d_{ij} < r) = 1$ if the distance d_{ij} between i and j is $\leq r$, otherwise $I(d_{ij} < r) = 0$. The estimated K-function is usually compared with the value of a homogeneous planar Poisson process with the same density as the given distribution for which $K(r) = \pi r^2$. In this paper we denote with $K'(r)$ the difference between the estimated value $\hat{K}(r)$ and value $K(r)$ of the corresponding Poisson process. If the distribution of the items is regular the value of K' is smaller than zero and if the items are clustered the value of K' is larger than zero.

Summary Function. For a spatial point process X let F be the distribution function of the distance from a fixed point in space to the nearest point of X. F is called the Empty Space Function. The Nearest Neighbour Distance Distribution Function G is the cumulative distribution function of the distance from a random point of X to the nearest other point of X. Similar as in case of Ripley's K-function the functions F and G can be estimated for a given distribution of items



(a) $\mathcal{S} = 10.97, E_5 = 13.00$ (b) $\mathcal{S} = 11.88, E_5 = 6.532063$ (c) $\mathcal{S} = 7.76, E_5 = 5.53$

Fig. 2. Clustering over time; no $\mathcal{AC}$-agents were used; depicted is the distribution of items after 100.000 (a), 1.000.000 (b), and 50.000.000 (c) simulation steps; hierarchical social entropy $\mathcal{S}$ and spatial entropy E_5 measures are given below the subfigures

in a cell array by functions $\hat{F}$, respectively $\hat{G}$. Functions $\hat{F}$ and $\hat{G}$ are often used for data analysis and usually provide a good statistic on the sizes of gaps in the pattern. The so-called summary function $\hat{J}(r)$ that was introduced in [18] is defined as

$$\hat{J}(r) = (1 - \hat{G}(r))/(1 - \hat{F}(r))$$

Again, similar to the case of Ripley's K-function, the estimations $\hat{F}$, $\hat{G}$, and $\hat{J}$ are often compared to a complete random point process with intensity λ, for which $F(r) = G(r) = 1 - exp(-\lambda \cdot \pi \cdot r^2)$ and $J(r) = 1$ holds. Therefore, a value of $\hat{J}(r) < 1$ indicates a clustered pattern, whereas a value of $\hat{J}(r) > 1$ can be interpreted that the pattern is ordered. For the computation of $\hat{J}(r)$ we used the corresponding function in the spatstat package ([2]).

Hierarchical Social Entropy In [3] Balch proposed the hierarchical social entropy measure. Let $\mathcal{R} = \{r_1, \ldots, r_N\}$ the items for which the measure is to be calculated. For each pair of items a distance $d(r_i, r_j)$ is given.

In the first step a hierarchical clustering is calculated as follows. Initially each item is assigned to its own cluster. Then iteratively the two most similar clusters are merged, until there is just one single cluster left. The distance between two clusters can be computed in different ways. We choose the so called complete linkage method. For this method the dissimilarity between 2 clusters is the maximal distance between two arbitrary items of both clusters. For more details on hierarchical clustering see for example [6].

The hierarchical clustering leads to a dendrogram which visualizes the agglomeration process in a binary tree, where the leaves of the tree are identified by the items to be clustered. Two nodes are siblings if their corresponding clusters are agglomerated during the hierarchical clustering. Note, that each inner node of the dendrogram corresponds to a taxonomic level h, i.e., the two clusters C_1 and C_2 have a dissimilarity of $D(C_1, C_2) = h$. For a given taxonomic level h the items of $\mathcal{R}$ are classified by the hierarchical clustering into clusters $\mathcal{C}(h) = \{c_1, \ldots, c_{M(h)}\}$. The hierarchical social entropy of a set of items $\mathcal{R}$ is defined as

$$\mathcal{S}(\mathcal{R}) = \int_0^\infty H(\mathcal{R}, h)dh,$$

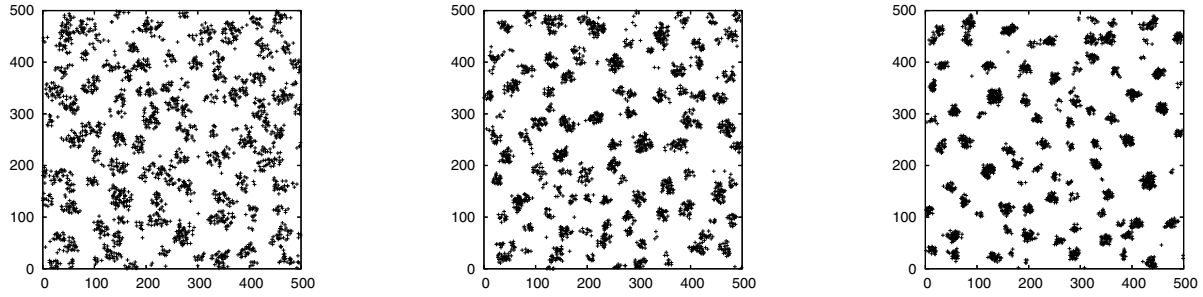
where $H(\mathcal{R}, h) = -\sum_{i=1}^{M(h)} p_i \log_2(p_i)$ is the simple social entropy of $\mathcal{R}$ at level h (p_i is the proportion of agents in the i-th subset c_i). The hierarchical social entropy enables a total ordering according to the diversity of situations where items are distributed in a (two-dimensional) space. Note, that the hierarchical social entropy is scale invariant and allows to address the extend of differences between clusters. In [3] the measure was used to calculate the diversity of a set of robots. In this paper we will use the measure to distinguish between fine grained and coarse grained clustering situations. In Figure 1 a clustering situation, the resulting dendrogramm, and the value of the social entropy at different taxonomic levels is depicted.

V. Results

If not stated otherwise a two-dimensional cell array of size 500×500 is used for the experiments. In the initial state 2500 cells ($1/100$ of all cells) are occupied by an item. The number of clustering agents was set to 50. The neighborhood of an agent is defined as the von Neumann neighborhood with radius 10, i.e. all cells for which $d_x + d_y \leq 10$ are said to be in the neighborhood of an agent, where d_x and d_y are the absolute distances of the considered cell to the cell of the agent in the two dimensions. In the following the results for different scenarios are presented. The threshold parameters for ant clustering were chosen as $k_1 = 0.05$ and $k_2 = 0.03$.

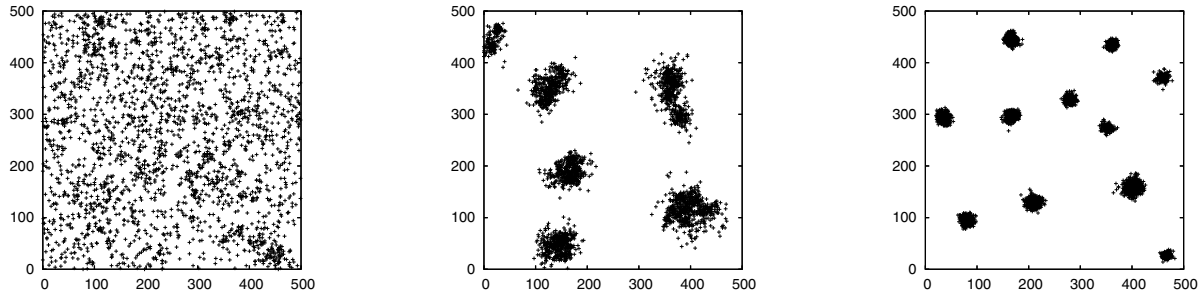
A. No $\mathcal{AC}$-Agents

For reasons of comparison the results for clustering without any anti-agents are presented first. In Figure 2 the clustering behavior after 100.000, 1.000.000, and 50.000.000 simulation steps (i.e., each of the agents performs so many simulation steps) can be seen. As expected it can be clearly seen that there is a clustering behavior over time. The corresponding hierarchical social entropy and the spatial entropy measures are given in the caption of the figure. The values for Ripley's K-function (more precisely the difference of the estimated K-function from a random point process) are depicted in Figure 6. It is interesting to point out an effect which could not be reflected in a one-valued validity measure: the maximal value for $K'(r)$ over time is shifted towards larger values of r. This corresponds to different strength of clustering on different



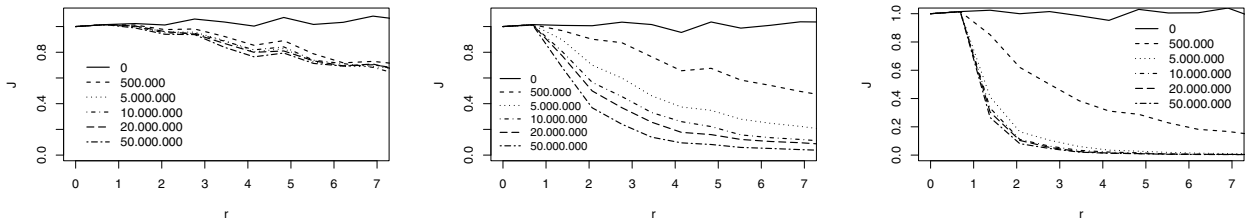
(a) $\mathcal{S} = 12.60, E_5 = 7.39$ (b) $\mathcal{S} = 13.00, E_5 = 7.16$ (c) $\mathcal{S} = 11.67, E_5 = 6.82$

Fig. 3. Clustering over time; different numbers of reverse $\mathcal{AC}$-agents together with 50 standard agents after 1.000.000 steps; (a) 5000 $\mathcal{AC}$-agents, (b) 1000 $\mathcal{AC}$-agents, (c) 100 $\mathcal{AC}$-agents



(a) $\mathcal{S} = 13.64, E_5 = 7.56$ (b) $\mathcal{S} = 8.07, E_5 = 6.73$ (c) $\mathcal{S} = 6.91, E_5 = 5.78$

Fig. 4. Clustering over time; different numbers of random $\mathcal{AC}$-agents together with 50 standard agents after 50.000.000 steps; (a) 100 $\mathcal{AC}$-agents, (b) 50 $\mathcal{AC}$-agents, (c) 10 $\mathcal{AC}$-agents



(a) 100 random $\mathcal{AC}$-agents (b) 50 random $\mathcal{AC}$-agents (c) 10 random $\mathcal{AC}$-agents

Fig. 5. Clustering over time; random $\mathcal{AC}$-agents together with 50 standard agents; depicted is the Summary Function $J(r)$ for different time steps for the test runs for which the final clustering situation is shown in Figure 4

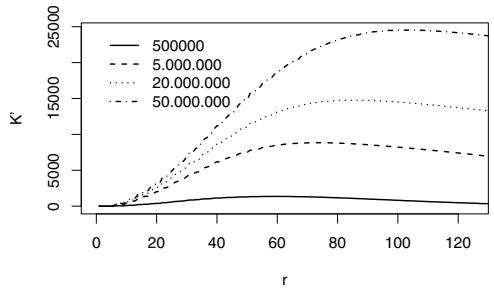


Fig. 6. K'(r) after different number of iteration steps; no $\mathcal{AC}$-agents were used; (cmp. Figure 2)

scales, because it means that the deviation from the expected number of points within a circle with radius r for a random point distribution is maximal for different values of r. Over time the cluster sizes increase, and therewith also the peak of $K'(r)$. After 500.000 steps the maximal value for $K'(r)$ can be found at $r \approx 60$ and after 50.000.000 simulation steps at $r \approx 100$. Note, that due to border effects the value of $K'(r)$ for large values of r may be misleading.

B. Reverse $\mathcal{AC}$-Agents

The influence of reverse $\mathcal{AC}$-agents (we describe the results for the type 1 agents only because the results of type 2 agents are very similar) is shown in Figure 3 where the distribution of the items after 1.000.000 simulation steps is shown for different number of reverse $\mathcal{AC}$-Agents. It can be seen that it

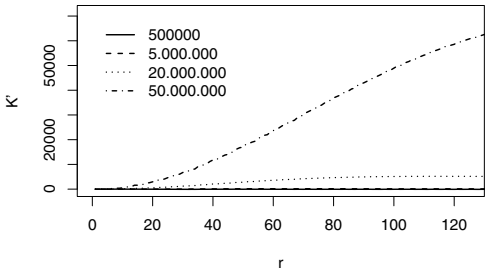


Fig. 7. K'(r) over time for 35 deterministic $\mathcal{AC}$-agents together with 50 standard agents; (cmp. Figure 8)

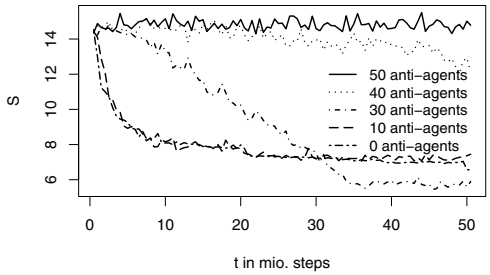


Fig. 10. Hierarchical social entropy $J(r)$ for different number $k \in 0, 10, 30, 40, 50$ of deterministic $\mathcal{AC}$-agents together with 50 standard agents

is not possible for the $\mathcal{AC}$-Agents to hinder the standard agents to perform a clustering - even if 100 times more reverse $\mathcal{AC}$-agents are used the item distribution is similar as for the only standard agents after $1.000.000$ simulation steps (cmp. Figure 2b)) . The differences between using 100 or 5000 reverse $\mathcal{AC}$-Agents are small. For the latter the clusters are more diffuse but the number of clusters is similar is nearly the same. This results show that it is not a trivial task to find efficient anti-clustering agents.

C. Random $\mathcal{AC}$-Agents

The results for random $\mathcal{AC}$-agents are depicted in figures 4 and 5. Figure 4 shows that no strong clustering occurs for 100 random $\mathcal{AC}$-Agents. Thus, a reasonable number of random $\mathcal{AC}$-Agents are able to hinder the clustering and are therefore possible candidates to be used as anti-clustering agents.

But it can also be seen that using a medium number of 50 random $\mathcal{AC}$-agents even enhances the degree of clustering compared to using less (0 or 10) or more 100 random $\mathcal{AC}$-agents. Hence, the strength of clustering can even improve with respect to using only standard clustering ants when a certain number of random $\mathcal{AC}$-Agents is used. Even if the clusters are slightly more diffuse, their number is clearly smaller than using no ant-clustering ants. One consequence of the more diffuse clusters is that more standard agents carry items in later phases of the clustering. The result is that smaller clusters are dissolved faster. It should be noted that this result is very interesting for ant clustering algorithms in general because it shows that the addition of agents which have the effect to make clusters more diffuse can lead to improved clustering methods. This aspect has never been studied before.

Clearly, what a good clustering method is depends on how clustering is defined. For the Summary Function $J(r)$ it can be seen that the quality of clustering decreases with an increasing number random $\mathcal{AC}$-agents. Figure 5 shows that no ordered item pattern occur (the value $J(r)$ is always smaller than 1) and that, as expected, the more $\mathcal{AC}$-agents are used, the longer it takes until a certain degree of clustering occurs. This can be seen when comparing the curves for the same simulation step of the three subfigures in Figure 5. 100 random $\mathcal{AC}$-Agents even prevent a good clustering. Note, that the hierarchical social entropy measure shows clearly different values for the case of using 50 or 100 random $\mathcal{AC}$-agents.

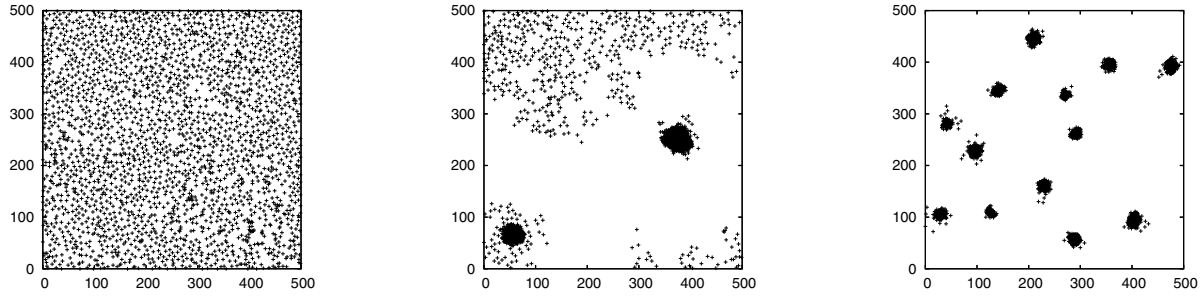
D. Deterministic $\mathcal{AC}$-Agents

The most interesting type of $\mathcal{AC}$-agents are deterministic $\mathcal{AC}$-agents which have a deterministic picking and dropping behavior. Figure 8 shows the item distribution after 50 million steps for different number of anti-clustering agents. It can be seen that 50 $\mathcal{AC}$-agents clearly hinder the standard ants to perform a successful clustering. To confirm this statement we also performed tests where the initial situation was a clustered situation. In this case the $\mathcal{AC}$-agents destroyed the clustering successfully. Hence, the deterministic $\mathcal{AC}$-agents can be called efficient because they "win" against the same number of clustering agents.

A more detailed analysis is shown in Figure 9, in which the Summary Function $J(r)$ is depicted. In can be seen clearly in Subfigure 9(a) that for 50 deterministic $\mathcal{AC}$-Agents over time ordered patterns occur (i.e., no clustering occurs). Note, that ordered patterns have been observed only for this type of $\mathcal{AC}$-agents. Using only 10 $\mathcal{AC}$-agents can not hinder the clustering agents to perform their task successfully (cmp. Figure 8(c) and 9(c)).

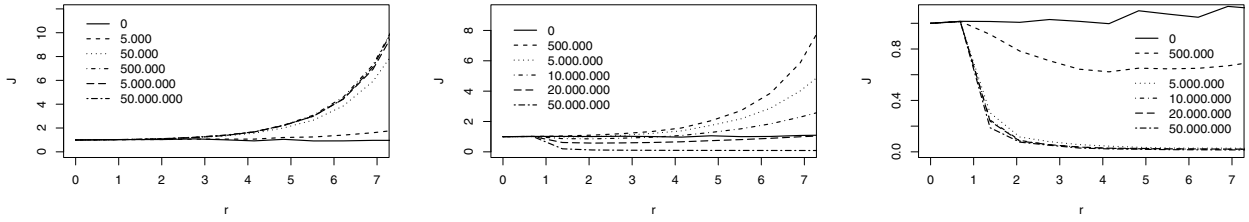
An interesting behavior can be observed when using a medium number of 35 deterministic $\mathcal{AC}$-agents. Figure 9(b) shows, that after 500.000 steps ordered patterns occur, i.e., the $\mathcal{AC}$-agents clearly prevent a clustering. But for an increasing number of steps, the ordered pattern disappears and a clustering occurs. After 20 million steps the value of the Summary Function radius of 2 is $J(r) \approx 0.5$ but for large radiuses ($r = 7$) $J(r) \approx 0$. This is very interesting, because over the time a situation may occur, where there is an ordering of items on a large scale, but a clustering on a small scale. Moreover it shows that a system with a combination of agents that when alone show an emergent behavior together with anti-emergence agents can show a very complex behavior. The increase of ordered patterns that prevents the emergence behavior might exists only over certain time periods before the ordered pattern brakes down and the emergent behavior can appear.

Similar as for the random $\mathcal{AC}$-agents, a medium number of deterministic $\mathcal{AC}$-agents even supports the clustering agents in their task. This can be clearly seen in Figure 10 where the values of the hierarchical social entropy $\mathcal{S}$ are shown over time for different numbers of deterministic $\mathcal{AC}$-agents. 50 agents were able to hinder the clustering agents ($\mathcal{S} > 14.0$), when



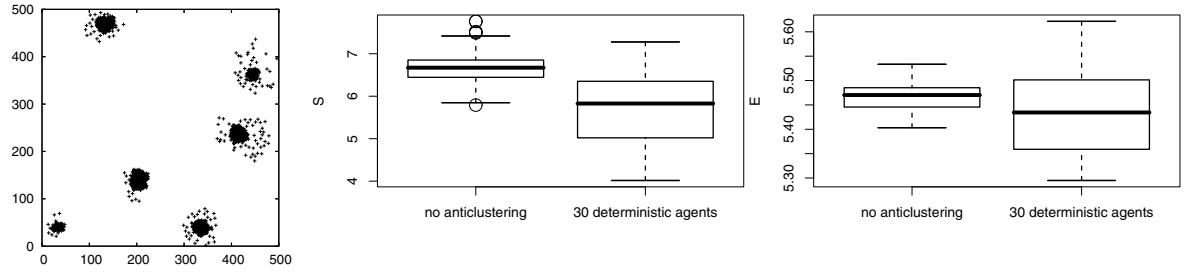
(a) $\mathcal{S} = 14.94, E_5 = 7.75$ (b) $\mathcal{S} = 5.96, E_5 = 5.86$ (c) $\mathcal{S} = 7.45, E_5 = 5.59$

Fig. 8. Clustering over time; deterministic $\mathcal{AC}$-agents together with 50 standard agents after 50.000.000 steps; (a) 50 $\mathcal{AC}$-agents, (b) 35 $\mathcal{AC}$-agents, (c) 10 $\mathcal{AC}$-agents



(a) 50 deterministic $\mathcal{AC}$-agents (b) 35 deterministic $\mathcal{AC}$-agents (c) 10 deterministic $\mathcal{AC}$-agents

Fig. 9. Clustering over time; different numbers of deterministic $\mathcal{AC}$-agents together with 50 standard agents; depicted is the Summary Function $J(r)$ for different time steps for the test runs for which the final clustering situation is shown in Figure 8



(a) (b)

Fig. 11. Subfigure (a): clustering situation after 50 million steps when using 30 deterministic $\mathcal{AC}$-agents together with 50 standard agents; Subfigure (b): Hierarchical social entropy $\mathcal{S}$ (left) and spatial entropy measure E_5 (right) after 50 million simulations steps when using 0 or 30 deterministic $\mathcal{AC}$-agents together with 50 standard agents

using no $\mathcal{AC}$-agents a clustering with $\mathcal{S} \approx 7$ was achieved, and interestingly 30 $\mathcal{AC}$-agents supported a final clustering with $\mathcal{S} \approx 6$. Also the $K'(r)$ value for the clustering is much larger when using 35 $\mathcal{AC}$-agents. This is depicted in Figure 7. After 50 million steps it holds that $K'(r) \approx 50.000$, whereas when using no $\mathcal{AC}$-agents a value of $K'(r) \approx 25.000$ was achieved.

To show that the hierarchical social entropy measure $\mathcal{S}$ can better differentiate between situations with several clusters both measures have been compared. 25 runs using no $\mathcal{AC}$-agents and 25 runs using 30 deterministic $\mathcal{AC}$-agents have been performed. Figure 2(c) (11(a)) shows the results of one of those runs for no $\mathcal{AC}$-agent (respectively 30 deterministic $\mathcal{AC}$-agents). It can be seen that the clusters are more clear without the $\mathcal{AC}$-agents. Figure 11(b) shows boxplots for the spatial entropy and the hierarchical social entropy. The spatial entropy measure is not able to differentiate between the two situations whereas the hierarchical social entropy measure shows that the clustering is stronger when no $\mathcal{AC}$-agents are used (Note, that also different patch sizes have been tried, results are not given here). A Wilcoxon Signed-Rank Test shows a significant difference for the values of the hierarchical social entropy, but not for the spatial entropy measure (a p-value of 0.01 was used).

E. Summary

We want to stress three interesting findings from the presented results. i.) for the random and the deterministic $\mathcal{AC}$-agents there is a threshold value for the number of $\mathcal{AC}$-agents, i.e. a degree of anti-emergence influence, that hinders the clustering agents to perform their task successfully. ii) Using

a smaller number of $\mathcal{AC}$-agents may even help the standard agents to cluster faster. iii.) The combination of $\mathcal{AC}$-agents and standard agents may lead to situations which have a ordered pattern on a large scale, and a clustered pattern on a small scale. This can not be reflected in one-valued measures like the hierarchical social entropy or the spatial entropy.

VI. Conclusions

This study was motivated by the problem how negative emergence can be prevented in autonomous and organic computing systems. Existing methods rely at least partly on classical control theory and therefor use principles which restrict the autonomy of the controlled system components. Here we have introduced a new approach that was called Swarm Controlled Emergence. The main idea of this approach is to introduce a swarm of anti-emergence components (or $\mathcal{AC}$-agents) into the system which prevent the negative emergent behavior. Ideally, this anti-emergence components should be similar to the normal components as far as possible. As an example system the well known emergent clustering behavior of ants was used. Three different types of anti-clustering agents have been investigated for this system. Namely, reverse, random, and deterministic $\mathcal{AC}$-agents. The reverse $\mathcal{AC}$-agents could not prevent the clustering process which shows that simply reversing the standard agents behavior is not enough to prevent the clustering. Both the random and the deterministic $\mathcal{AC}$-agents could successfully prevent the clustering with a moderate number of individuals. We could also show the interesting effect that introducing a medium (but smaller than necessary to prevent the clustering) number of these agents can even increase the strength of the clustering. This observation can be used for the design of clustering algorithm and might lead to better ant clustering algorithms. It was shown that the combination of clustering and anti-clustering agents can lead to a system that prevents clustering over some time period (and even leads to the emergence of ordered patters) after which the order vanishes and a clustering appears. This study is only a first step to systems that use Swarm Controlled Emergence and it will be interesting for future research to design and analyze other such systems.

Acknowledgment

This work was supported by the German Research Foundation (DFG) through the project Organisation and Control of Self-Organising Systems in Technical Compounds within SPP 1183.

References

[1] GI: Organic Computing / VDE, ITG, GI - Positionspapier. 2003, online: http://www.betriebssysteme.org/Betriebssysteme/FutureTrends/oc-positionspapier.pdf

[2] A. Baddeley and R. Turner: `spatstat` website, URL: `www.spatstat.org`

[3] T. Balch: Hierarchical Social Entropy: An Information Theoretic Measure of Robot Team Diversity. Autonomous Robots, 8(3): 209–237 (2000).

[4] E. Bonabeau and M. Dorigo, and G. Theraulaz: Swarm Intelligence: From Natural to Artifi cial Systems. Oxford University Press, New York, (1999).

[5] L. Chrétien: Organisation spatiale du matériel provenant de l'excavation du nid chez Messor barbarus et des cadavres d'ouvrières chez Lasius niger (Hymenopterae: Formicidae). Université Libre de Bruxelles (1996)

[6] W. H. E. Day and H. Edelsbrunner: Effi cient algorithms for agglomerative hierarchical clustering methods. Journal of Classifi cation, 1(1): 7–24 (1984).

[7] J.L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, L. Chretien: The dynamics of collective sorting: robot-like ants and ant-like robots. In: J.-A. Meyer et al. (Eds.) Proc. Simulation of Adaptive Behavior: From Animal to Animats, 356–365 (1991).

[8] N.R. Franks and A. Sendova-Franks: Brood sorting by ants: distributing the workload over the work surface. Behav. Ecol. Sociobiol, 30:109–123 (1992).

[9] H. Gutowitz: Complexity Seeking Ants. Unpublished report, (1993).

[10] J. Handl, B. Meyer: Improved Ant-based Clustering and Sorting in a Document Retrieval Interface. Proc. Seventh Int. Conference on Parallel Problem Solving from Nature (PPSN VII). LNCS 2439, 913–923 (2002).

[11] J. Handl, J. Knowles, and M. Dorigo: Strategies for the increased robustness of ant-based clustering. Postproc. of the First International Workshop on Engineering Self-Organising Applications (ESOA 2003), LNCS 2977, 90–104 (2003).

[12] J. Handl, J. Knowles, and M. Dorigo: On the performance of ant-based clustering. In: Proc. 3rd Int. Conf. on Hybrid Intell. Systems (HIS 2003), IOS Press, (2003).

[13] P. M. Kanade, L. O. Hall: Fuzzy Ants as a Clustering Concept. Proceedings 22nd International Conference of the North American Fuzzy Information Processing Society NAFIPS, 227–232 (2003).

[14] P. Kuntz, D. Snyers: Emergent colonization and graph partitioning. In: D. Cliff et al. (Eds.), Third Int. Conf. on Simulation of Adaptive Behavior: From Animals to Animats, MIT Press, 494–500 (1994).

[15] N. Labroche, N. Monmarche, G. Venturini . A new clustering algorithm based on the chemical recognition system of ants. Proc. European Conf. on AI, IOS Press, 345–349 (2002).

[16] N. Labroche, N. Monmarche, G. Venturini: AntClust: Ant Clustering and Web Usage Mining. Proc. of GECCO-2003, Springer, LNCS 2723, 25–36 (2003).

[17] N. Labroche, N. Monmarche, G. Venturini. Visual clustering with artifi cial ants colonies. Proc. 7th International Conference on Knowledge-Based Intelligent Information & Engineering Systems (KES 2003), Springer, LNCS 2773, 332–338 (2003).

[18] M.N.M. van Lieshout and A.J. Baddeley: A nonparametric measure of spatial interaction in point patterns. Statistica Neerlandica, 50:344–361 (1996).

[19] D. Merkle and M. Middendorf and A. Scheidler: Modelling Ant Brood Tending Patterns with Cellular Automata. Journal of Cellular Automata, 2(2):183-194 (2006).

[20] M. Mnif and C. Müller-Schloer: Quantitative Emergence. Proc. of the 2006 IEEE Mountain Workshop on Adaptive and Learning Systems (IEEE SMCals 2006), 2006.

[21] C. Müller-Schloer, C. von der Malsburg and R. P. Würtz: Organic Computing. Informatik Spektrum, 27(4):332–336, 2004

[22] C. Müller-Schloer, B. Sick: Emergence in Organic Computing Systems: Discussion of a Controversial Concept. Proc. 3rd International Conference on Automic and Trusted Computing, Springer, LNCS 4158, 1–16, 2006.

[23] U. Richter, M. Mnif, Jürgen Branke, C. Müller-Schloer and H. Schmeck: Towards a generic observer/controller architecture for Organic Computing. Köllen Verlag, Lecture Notes in Informatics P-93, 112-119, 2006.

[24] B.D. Ripley: Modelling Spatial Patterns. Journal of the Royal Statistical Society, 39:172–212 (1977).

[25] F. Rochner, C. Müller-Schloer: Emergence in Technical Systems. it - Special Issue on Organic Computing, 47: 188-200, (2005).

[26] H. Schmeck: Organic Computing – A New Vision for Distributed Embedded Systems. Proc. of the Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC 2005), 201–203 (2005).

[27] A.B. Sendova-Franks and J.V. Lent: Random walk models of worker sorting in ant colonies. Journal of Theoretical Biology, 217: 255–274 (2002).

[28] T. Schöler, C. Müller-Schloer: An Observer/Controller Architecture for Adaptive Reconfi gurable Stacks. Proceedings ARCS 05, Springer, LNCS 3432, 139-153 (2006).

[29] G. Theraulaz, E. Bonabeau, S.C. Nicolis, R.V. Sole, V. Fourcassie, S. Blanco, R. Fournier, J.-L. Joly, P. Fernandez, A. Grimal, P. Dalle, J.-L. Deneubourg: Spatial Patterns in Ant Colonies. PNAS, 99: 9645–9649 (2002).