Towards an automated detection of self-organizing behavior

Wolfgang Trumler, Mike Gerdes
University of Augsburg
Eichleitnerstrasse 30
D-86159 Augsburg

$\{trumler|gerdes\}$@informatik.uni-augsburg.de

Abstract: In this paper we present an entropy based method to analyze complex systems. Systems are treated as black boxes, which only expose information by some specified parameters. To decide whether a system shows self-organizing behavior in terms of the specified parameters, the calculated entropy values are preprocessed using low-pass filters. The time derivative of the smoothed entropy curve is used to analyze the short-term behavior of the system. A larger filter is used for the analysis of the long-term behavior. Different evaluations show that in principle self-organizing behavior can be recognized with our approach.

1 Introduction

The major challenges for Organic Computing systems are to use the advantages of self-organization [Ash62][Sch97][Hey99][Sti98] by finding ways to recognize and control it [Sch05]. The first step is to monitor the system's parameters to extract information, which can be used as input to the controller. In many cases, especially for Organic Computing systems, no exact behavioral model of the overall systems exists, because these systems are intentionally built to expose self-organizing behavior. Thus, it is hardly possible to reason about the actual state of the system and when a change leads to self-organization or not. The same applies for distributed systems or systems which must be treated as a black-box, where the system state is unknown, but only some parameters can be measured.

Our work is a generic approach towards an automated detection of self-organizing behavior employing entropy values [EFS98] calculated from measured parameters. Different approaches are used to evaluate the short-term and long-term behavior. Before we calculate the time derivative of the entropy values we apply a low-pass filter to smooth the entropy values. Our approach analyzes the short-term as well as the long-term behavior of the entropy values, because only the combination of both gives meaningful information in terms of a system's self-organizing behavior. The output of our automated approach matches very closely the findings of scientists with appropriate domain knowledge.

The remainder of this paper is structured as follows. In the next section related work is presented. In Section 3 and 4 we present the details of our approach for an automated detection of self-organizing behavior. Section 5 shows the evaluation results of two examples and the paper closes with a conclusion in Section 6.

2 Related Work

In [MMS06] a controller/observer-architecture is introduced to determine emergent behavior of complex systems. In order to decide whether a system shows emergent behavior or not, the authors first need to show that the system is self-organizing, as they postulate self-organization as a precondition for emergent behavior. Therefore they calculate the entropy at the beginning and at the end of a process. If the difference is greater than zero the system is supposed to exhibit emergent behavior. The drawback of this approach is that the user has to manually choose the start and end point of the process used to calculate the difference of the two entropy values. Significant changes inside the interval are totally ignored. So, relevant changes indicating self-organizing behavior will be missed. Our evaluations of the same model [Ger08] show that we do not need to explicitly declare the start and end points because our system evaluates the entropy over the whole period of time. Furthermore we show that it is important to analyze the short-term and long-term behavior of the system to detect self-organizing behavior.

Our approach using entropy values to extract information from different parameters has strong similarities with decision trees, e.g. trees built by C4.5 algorithm [Qui93]. These algorithms use the entropy to calculate the information gain at every single time step, to find the most suitable parameter for the next split of the data set. We are not interested in finding a good split of the data set but in the chronological characteristic of the entropy values over the contemplated time interval.

3 Entropy value calculation

In general we treat complex systems as black boxes and only want to use the measured parameters of a system for our automated detection of self-organizing behavior. Therefore, we have to process the information with the following steps to finally calculate the entropy values. First the parameter values must be discretized to create a histogram in the second step. With the histogram the probabilities of every possible value can be calculated, which is needed for the calculation of the entropy values. As we want to compare the entropy values of different parameters, we calculate the normalized entropy values for every parameter. An entropy value has to be calculated for every simulation step. So, the calculations, which are explained in detail in the following sections, have to be done for every single simulation step of every parameter.

3.1 Value discretization

The calculation of an entropy value depends on a the amount of possible values within a parameters' value range. As we want to build a system that can be used at runtime to detect self-organizing behavior, we must restrict the amount of possible values of a parameter. This is done by discretizing real-valued parameters and discrete parameters with a high range of possible numbers.

For our approach we assumed an upper bound of 1000 discretization steps, which was never exceeded for any of the examples we evaluated. We know that we introduce a small error concerning the calculation of the entropy. On the other hand the discretization leads to much faster calculations and gives our system the capability to be used online.

3.2 Histogram creation and probability calculation

With the discretized parameter values we can create a histogram for every parameter. The histograms are needed for calculation of the probabilities of the parameter values. Therefore, we count the amount of every single parameter value in the input data.

In the evaluated chicken farm simulation (see Section 5) for example, we count the x- and y-coordinate of every chicken in two histograms. In the simulation data a field with 30 x- and y-positions were given. So, the histograms for the x- and y-coordinates had a size of 30. If n chickens were on the same field the x and y histograms would have the number n for one value and zero for all other 29 values.

If the histogram has n bins, the probability p_i of a value (bin) in the histogram is calculated with Equation 1 by dividing the amount of entries for one value m_i at position $1 \leq i \leq n$ in the histogram by the total amount of entries k. The probability at position i in the histogram is zero, if there is no entry at i and one, if only one position i of the histogram has all k entries in the histogram.

$$p_i = m_i/k \tag{1}$$

3.3 Entropy calculation

The automated analysis calculates the entropy values H [Sha48] of each parameter at every simulation step with Equation 2. The value p_i is the probability of a discretized value of the histogram as calculated in the previous section. The upper bound n in Equation 2 refers to the amount of bins in the previously calculated histogram.

$$H = -\sum_{i=0}^{n} p_i \cdot \log_2(p_i) \tag{2}$$

As we want to compare different data series we need to normalize the calculated entropy values. The normalized entropy value H_{norm} of each parameter is calculated as shown in Equation 3 using the maximum entropy H_{max}. The maximum entropy is reached if every position i of the histogram has the same amount of entries, which leads to the same probability p_i for all values of a parameter.

$$H_{norm} = \frac{H}{H_{max}} \cdot 100 \ , \ \ H_{max} = \log_2(n) \tag{3}$$

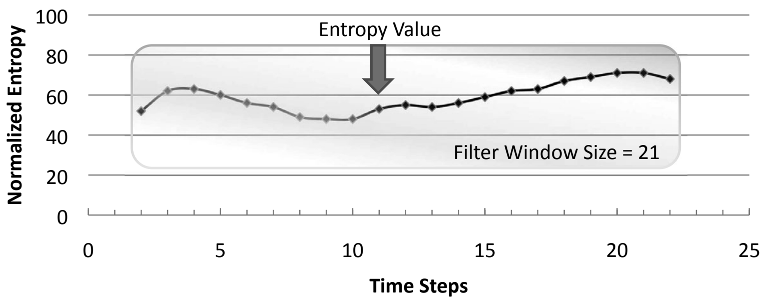


Figure 1: Rectangle filter used to smoothen the entropy values

4 Automated detection of self-organization

Different complex systems may show varying characteristics of self-organization, e.g. one system may change often and rather fast from one state to another and another system may change states slowly over a longer period of time. Thus, a combined analysis of the short-term and long-term behavior is used by our developed tool.

For the automated detection of self-organization, we need to post-process the calculated entropy values. Therefore, we use filters and the time derivative of the entropy values as input for the analysis of the systems' short-term and long-term behavior.

4.1 Filtering the entropy values

The entropy values may show large differences between two time steps, leading to a discontinuous curve, therefore a low-pass filter is used to smooth the entropy values. The low-pass filter calculates the mean of a specific range of entropy values specified by the filter window size as shown in Figure 1. We use a small window size (e.g. 25 time steps) for the calculation of the time derivative for the short-term behavior and a larger window size (e.g. 10% of all simulation steps) for the analysis of the long-term behavior. These filter window sizes can be changed manually with our tool, but the above values have shown good results in all of our evaluations.

4.2 Analyzing short-term and long-term behavior

The analysis of the short-term behavior employes the time derivative of the normalized and smoothed entropy values. The time derivative values are calculated with a rectangle filter as shown in Figure 2 with the same filter window size that is used to smooth the entropy values (e.g. 25 time steps).

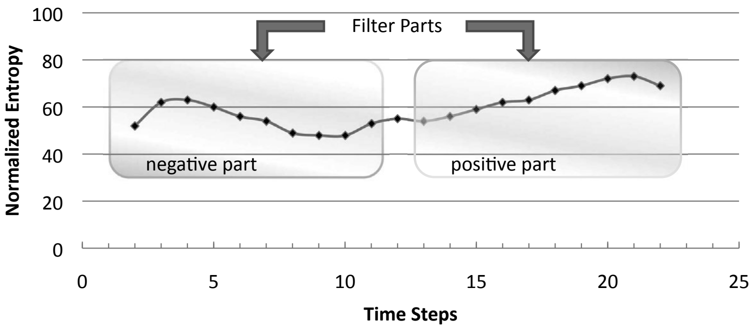


Figure 2: Rectangle filter used to calculate the time derivative of the entropy values

The short-term behavior analysis checks the whole time span to detect peaks and monotone regions in the time derivative. Depending on the maximum change of the time derivative in a monotone region, we classify this region in one of five classes, which vary from "very low change" to "very high change". We are especially interested in significant changes and at which simulation step they occur as they are indicators for self-organizing behavior.

Due to the fact that we normalize the entropy values in the range of 0 to 100, we were able to define fixed ranges for the five classes of the local behavior. It turned out that "very high changes" of the time derivative need to exceed a value of 15. The range from 0 to 15 is equally split among the four remaining classes starting with "very low change" from 0 to 3.75, up to "high change" in the range from 11.25 to 15. Nevertheless, the short-term behavior analysis fails to recognize a slowly changing system because all changes are categorized as "very low" in that case.

To recognize long-term changes in the entropy values, we use a global analysis. Therefore, the entropy values, are filtered with a low-pass filter of a larger window size (normally 10% of all time steps). We divide the range of the entropy values and the time steps into three equally sized intervals. Depending on the entropy level, which dominates a time interval, a system is classified to one of the three aforementioned entropy levels. If our system recognizes a change of the entropy level from one time interval to the next, this indicates a long-term self-organizing behavior.

5 Evaluation

To evaluate our automated analysis, we used different systems as input for our tool. At this point we present the results of two simulation models. The first one, the chicken simulation model, was developed at the University of Hanover [CMMS$^+$07] and used for their work to investigate an approach to a quantitative emergence measure. The simulation data consists of the x- and y-coordinates of chicken movements in a 2D grid. The negative self-organizing behavior in this scenario occurs if a chicken is injured and the other chickens hunt and surround the injured one and try to peck it to death.

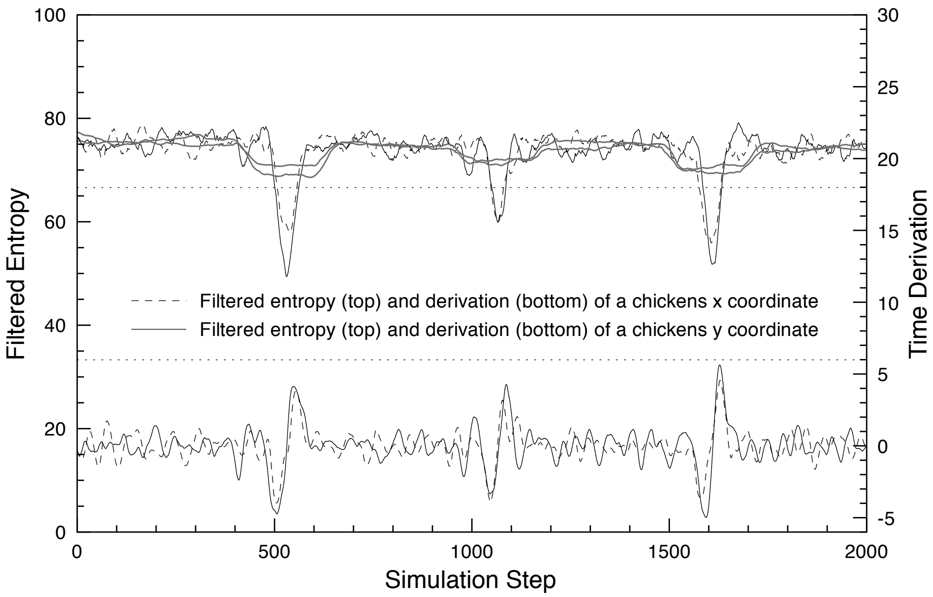
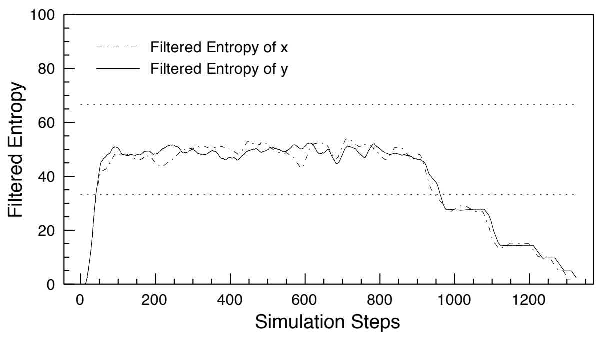


Figure 3: Entropy and time derivative for the chicken simulation model

For the evaluation we contemplate a simulation with 20 chickens moving in a grid of 30x30 fields over a period of 2000 simulation steps. The upper part of Figure 3 shows the filtered and normalized entropy values over the simulated time steps for the chickens' x- and y-coordinates (dashed and solid line). The strongly filtered entropy values (filtered with a large filter window size) are plotted in a paler gray on top of the other filtered entropy values (both correspond to the left y-axis). The lower part of Figure 3 shows the entropys' time derivative values with the y-axis on the right side.

The results of the long-term behavior analysis reveal no relevant details on a long-term change - the strongly filtered entropy values remain at a high level. But the analysis of the short-term behavior, based on the time derivative, identifies 5.2% lower and medium changes for the x-coordinates respectively 8.9% changes for the y-coordinates at simulation steps around 500, 1000 and 1500. The occurrence of an injured chicken is responsible for the self-organizing behavior, which leads to a decrease of the entropy values and the appearance of local minima, resulting in a corresponding zero-crossing in the time derivative values at that time step. The short-term analysis found those simulation steps automatically without further domain knowledge. In this case the experts with domain knowledge tell us that each 500 time steps an injured chicken is injected, which is exactly the outcome of the automated analysis.

Further evaluations of simulation data from this model showed [Ger08] that this system in general is changing quickly and that we need to cover the whole time span to detect the self-organizing behavior. If we just observe the start and the end, or some arbitrary



Global evaluation results:

Degree of Freedom 'x-coordinate':
Behaviour starting region:
medium distribution (75.94%)
Behaviour middle region:
medium distribution (88.29%)
Behaviour ending region:
low distribution (100.00%)

Degree of Freedom 'y-coordinate':
Behaviour starting region:
medium distribution (77.44%)
Behaviour middle region:
medium distribution (89.17%)
Behaviour ending region:
low distribution (100.00%)

Figure 4: Entropy and time derivative for the pollination model and results of automated global analysis

points of time to calculate the entropy difference as done in [MMS06], we would miss the interesting points.

A totally different simulation model of a self-organizing system is the pollination model [KB06]. It simulates the movements of bees while collecting and spreading pollen from flowers. The parameters consist of the x- and y-coordinates of bees, starting from one point in a grid (the hive), heading towards the flowers and returning to the starting point.

Figure 4 illustrates the filtered entropy values for 20 bees during 1326 simulation steps. The results of the global analysis show a change from medium distribution in the beginning (simulation steps 1 to 442) to a medium distribution in the middle region (simulation steps 443 to 884) to low distribution at the end of the simulation (simulation steps 885 to 1326). So the long-term behavior analysis recognizes this system behavior, especially the lowering of the entropy values from the middle to the end region, but fails to recognize the increase in the starting region due to the fast rise of the entropy values.

Although the short-term behavior analysis shows no relevant details in the middle of the simulation, it detects the change from a low entropy value at the beginning to the medium value after about 100 simulation steps. Combining the results from the global and the local analysis, we are able to recognize the fast change at the beginning of the simulation, the lowering at the end, as well as the system's long-term behavior. So, our combined approach allows us to recognize the self-organizing behavior of this simulation model. Although we are not able to recognize what kind of self-organizing behavior occurs, as this would require domain knowledge, we are able to recognize it anyhow. Including domain knowledge, the rise of entropy at the beginning of the data series characterizes the start of the bees from the hive, heading towards the flowers. In the end region the lowering of the entropy values results from the bees returning to the hive when their capacity is exhausted. As not all bees return at the same time the entropy values decrease stepwise.

6 Conclusion

Our system is a generic approach for an entropy based analysis, which uses the combined and automated short-term and long-term behavior analysis of entropy values over a period of time to characterize and examine self-organizing behavior of complex systems. The results of the evaluated systems indicate that this is a suitable approach as the combined use of the short-term and long-term behavior analysis works for different types of systems. Particularly, as we do not have any a-priori knowledge about the system behavior, we showed that this approach is capable of analyzing a system's behavior without any additional information despite the raw data from the measured parameters, which we use for our automated analysis.

References

[Ash62] Ross W. Ashby. Principles of the self-organizing system. *Transactions of the University of Illinois Symposium*, pages 255–278, 1962.

[CMMS$^+$07] E. Cakar, M. Mnif, C. Müller-Schloer, U. Richter, and H. Schmeck. Towards a Quantitative Notion of Self-organisation. In *Proceedings of the 2007 IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 4222–4229, Singapore, Sep. 2007.

[EFS98] Werner Ebeling, Jan Freund, and Frank Schweitzer. *Komplexe Strukturen: Entropie und Information*. Teubner, 1998.

[Ger08] Mike Gerdes. Diplomarbeit, Entwicklung eines Mechanismus zur Erkennung von Selbstorganisation. *University of Augsburg*, March 2008.

[Hey99] Francis Heylighen. The Science of Selforganization and Adaptivity. In *Computational and Mathematical Theory of Organizations*, volume 5(3), pages 253–280. Eolss Publishers, Oxford, UK, 1999.

[KB06] Holger Kasinger and Bernhard Bauer. Pollination - A Biologically Inspired Paradigm for Self-Managing Systems. *ITSSA*, 2(2):147–156, 2006.

[MMS06] Moez Mnif and Christian Müller-Schloer. Quantitative Emergence. In *Proceedings of the 2006 IEEE Mountain Workshop on Adaptive and Learning Systems (SMCals 2006)*, pages 78–84, Pacificaway, NJ, USA, July 2006. IEEE.

[Qui93] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[Sch97] Frank Schweitzer. Selbstorganisation und Information. *Komplexität und Selbstorganisation - Chaos in Natur- und Kulturwissenschaften*, pages 99–129, 1997.

[Sch05] H. Schmeck. Organic Computing - A New Vision for Distributed Embedded Systems. In *Proceedings of the Eighth IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, pages 201–203. IEEE Computer Society, 2005.

[Sha48] Claude Elwood Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, July, October 1948.

[Sti98] Ulf Stirke. *Technologie und Selbstorganisation - Zum Problem eines zukunftsfähigen Fortschrittsbegriffs*. PhD thesis, Universität Hamburg, Februar 1998.