Array 11 (2021) 100069

Contents lists available at ScienceDirect

Array

journal homepage: www.elsevier.com/journals/array/2590-0056/open-access-journal

# On the stability of (self-)adaptive behaviour in continuously changing environments: A quantification approach

Martin Goller, Sven Tomforde *

*Christian-Albrechts-Universität zu Kiel, Intelligent Systems, Hermann-Rodewald-Str. 3, 24118, Kiel, Germany*

ARTICLE INFO

*Keywords:*
Self-awareness
Self-adaptive and self-organising systems
System analysis
Stability
Degree of self-adaptation
Autonomic computing
Organic computing

ABSTRACT

The concept of self-adaptation and self-organisation (SASO) is a modern approach to cope with the ever-increasing complexity and interconnectedness of large-scale component systems. The basic idea is to react to environmental dynamics and disturbances by re-configuring the productive behaviour and/or the relations to other systems. However, this may result in unstable and even oscillating macro-level behaviour, potentially rendering the adaptation efforts of the contained component systems inappropriate. We assume that such an unstable configuration is an indicator of unexpected behaviour which can lead to a reduced utility of the overall system. To enable the system to be self-aware about such events, we propose a concept to measure the configuration stability of a SASO system by creating a derived time series based on the configurations. This is based on the application of the Kinoshita measure. We show the applicability of the concept and the observed behaviour in different simulated use-cases.

## 1. Introduction

Within the last decade, (self-)adaptivity increasingly became an integral part of the design of next-generation information and communication systems [1]. Concepts such as intelligent systems [2], cyber-physical systems [3], Internet of Things [4] or self-aware computing systems [5] established a trend in systems engineering towards constellations of typically distributed autonomous subsystems that integrate dynamically into an overall system constellation and self-adapt their behaviour in response to changing situations [6]. We refer to systems with abilities to autonomously modify their behaviour and their structural integration in terms of cooperation with other (sub-)systems as self-adaptive and self-organising (SASO) systems in the remainder of this article.

To equip a SASO system with the capability of context-aware self-adaptation, different approaches have been proposed [9], e.g. by the Automatic Computing [7] or Organic Computing [8] initiatives. gives an overview. These system designs often incorporate autonomous learning techniques to allow a continuous and unsupervised improvement of the self-adaptation behaviour [10]. The result of these efforts is that systems can react fast and continuously to changing conditions as well as disturbances [11] and consequently achieve higher robustness [12].

As outlined in previous work [13], we assume that too frequent adaptation decisions of the (sub-)systems will lead to decreased stability of the overall system and potentially to a decreased user acceptance. Although there might be a perfect reason for each decision about self-adaptation at sub-system-level, the resulting macro-level behaviour and the interplay of the distributed subsystems' behaviour may lead to unstable or even oscillating conditions. Although there is some work on the quantification of system properties and runtime behaviour in SASO system (see section 2.2 for an overview), what is missing is an integrated approach to balancing the adaptation decision between possible performance and robustness gains on the one hand and system stability and user acceptance on the other. With this article, we propose a first step towards such a constructive trade-off – the basis for this is an awareness of the actual stability within the system federation, based on external observation.

Based on the ideas to quantify a 'degree of adaptation', the authors have presented a first concept for measuring the stability of the system's configuration in Ref. [14]. The previous article evaluated the measure in an external disturbance scenario and a system failure scenario. This article extends the previous work by the evaluation of the measure in two new scenarios (see sections 4.2 and 4.4), determines novel insights about the behaviour and its interpretability, and gives an approach to determine the necessary parameters for the evaluation (see section 3.3).

The remainder of this article is organised as follows: Section 2

* Corresponding author.
*E-mail address:* st@informatik.uni-kiel.de (S. Tomforde).

https://doi.org/10.1016/j.array.2021.100069
Received 7 December 2020; Received in revised form 22 March 2021; Accepted 10 May 2021
Available online 28 May 2021

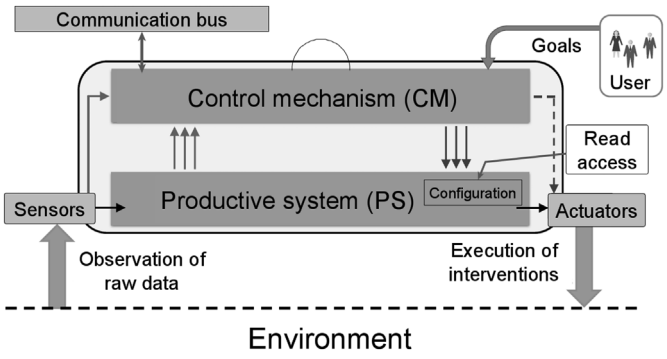


**Fig. 1.** Schematic illustration of a subsystem $a_i$ from Ref. [15]. The arrows from the sensors and PS to the CM indicate observation flows, while the arrows from the CM to the PS and the actuators indicate control flows. Dashed arrows emphasise a possible path that is typically not used. Not shown: The CM is able to communicate with other CMs in the shared environment to exchange information such as sensor reading and to negotiate policies.

explains the underlying system model for SASO systems including the assumptions made by the authors and describes related work. Section 3 presents an approach to the measurement of adaptation behaviour stability, which is experimentally analysed in section 4. Finally, section 5 summarises the paper and gives an outlook on future work.

## 2. Background

In this section, we initially describe our system model that defines which characteristics and capabilities of a SASO system we argue. Afterwards, we briefly summarise developments of related work.

### 2.1. System model

In this article, we refer to a SASO system $S$ as a collection $A$ of autonomous subsystems $a_i$ that are able to adapt their behaviour based on self-awareness of the internal and external conditions. We further assume that such a subsystem is an entity that interacts with other entities, i.e., other systems, including hardware, software, humans, and the physical world with its natural phenomena. These other entities are referred to as the *environment* of the given system. The *system boundary* is the common frontier between the system and its environment.

Each $a_i \in A$ is equipped with sensors and actuators (both, physical or virtual). Internally, each $a_i$ consists of two parts: The productive system part *PS*, which is responsible for the basic purpose of the system, and the control mechanism *CM*, which controls the behaviour of the PS (i.e., performs self-adaptation) and decides about relations to other subsystems. In comparison to other system models, this corresponds to the separation of concerns between *System under Observation and Control* (SuOC) and *Observer/Controller* tandem [15] in the terminology of Organic Computing (OC) [16] or *Managed Resource* and *Autonomic Manager* in terms of Autonomic Computing [7]. Fig. 1 illustrates this concept with its input and output relations. The user describes the system purpose by providing a utility or goal function $U$ which determines the behaviour of the subsystem. The User usually takes no further action to influence the decisions of the subsystem. Actual decisions are taken by the productive system and the CM based on the external and internal conditions and messages exchanged with other subsystems. We model each subsystem to act *autonomously*, i.e., there are no control hierarchies in the overall system. Please note that for the context of this article an explicit local configuration of the PS is necessary – which in turn limits the scope of the applicability of the proposed method. Furthermore, each subsystem must provide a read access to the configuration.

At each point in time, the productive system of each $a_i$ is configured using a vector $c_i$. This vector contains a specific value for each control variable that can be altered to steer the behaviour, independently of the particular realisation of the parameter (e.g., as real value, boolean/flag, integer or categorical variable). Each subsystem has its own configuration space, i.e. an n-dimensional space defining all possible realisations of the configuration vector. The combination of the current configuration vectors of all contained subsystems of the overall system $S$ defines the joint configuration of $S$. We assume that modifications of the configuration vectors are done by the different *CM* only, i.e. locally at each subsystem, and are the result of the self-adaptation process of the *CM*.

### 2.2. Related work

Within the last two decades, some work has been presented that aims at measuring and assessing SASO system properties. The most prominent examples are self-organisation [17] and emergence [18]. However, most of these metrics require domain knowledge, are dedicated to a specific application, are restricted to simple, small-scale models [19], or describe architectural and implementational properties for the comparison of systems at design-time [20].

Kaddoum et al. [21] discuss the need to refine classical performance metrics to SASO purposes and present specific metrics for self-adaptive systems. They distinguish between "nominal" and "self-*" situations and their relations. For instance, they measure the operation time about the adaptation time to determine the effort. Some of the developed metrics have been investigated in detail by Camara et al. for software architecture scenarios [22]. Besides, success and adaptation efforts and ways to measure autonomy have been investigated, see e.g. [23].

Only a few contributions focused on general metrics to determine the effort and the benefit of (self-)adaptation in distributed collections of autonomous subsystems, see Ref. [24] for an overview. Examples include the relation between working and adaptation time, the availability of subsystems for task processing, and the performance of the overall system (i.e., the degree to which a certain goal is achieved). This has been accompanied by a transfer of traditional performance metrics, see Ref. [21] or augmented with measures necessary for control strategies of SASO systems [25]. However, the focus on an degree of adaptation, the stability of adaptations, and the foundation for deriving a trade-off between possible performance gain of individual component systems and negative impact on macro-level behaviour has not been focused.

In general, there is only a very limited number of contributions on when (and if at all) to adapt in literature. Recently, Chen et al. presented a concept that is based on the so-called 'technical debt' [26]: The authors introduce a gate that either enables or disables the adaptation mechanism. This decision is based on deriving an integrated score for defining the current so-called 'temporal interest' and the expected 'revenue'. This implicitly allows for a similar goal as outlined in this paper: More stability rather than frequent self-adaptation. However, it does not come with a measurement that determines the degree of self-adaptation or the degree of stability, preferable in relation to the current context.

In [17] Tomforde et al. introduce a method to measure self-organisation based on a comparison of observed communication patterns between subsystems. Their basic idea is comparable to the approach presented in this paper. In Refs. [13,27] the authors present the idea to consider the configurations of subsystems as an observable state in a generative probabilistic model of observation. Although the technical approach is comparable to this article, the focus is different as it goes beyond a 'degree of self-adaptation/self-organisation'. Fundamentally, this article presents the next step on-top of these measurements.

As a summary of this discussion, we can state that a unified measurement framework for the quantification of externally measurable system properties that consider the adaptation behaviour of SASO systems is still missing. Either existing approaches focus on isolated aspects such as performance or they make use of domain knowledge. Based on the assumption of being able to assess the configuration of the productive components that are steered by adaptation modules, we add a novel approach for modelling system behaviour at macro-level. This serves as a starting point for higher-levelled assessments such as stability (discussed

in this article), variability, or acceptability (which is subject to future work). Technically, our probabilistic approach differs fundamentally from other approaches as it models each aspect of the distributed subsystems' configurations as random variables without any assumptions on the impact of the performance, the causes for adaptation decisions, or temporal implications. We compare the observations using divergence measures, which then allows for defining different measures for self-adaptation on-top of this quantification.

## 3. Configuration stability

If the CM decides that an adaptation is necessary it will change the configuration of the productive system. Since the decision models used by the CM are hidden to external observers, we model the configuration changes as a random process with the configuration vector as a random variable. Therefore, the observed configurations form a random distribution. For this distribution, an estimated density can be associated and new configurations can be assigned a probability concerning that density.

Our approach is based on the idea that if new configurations with a high probability are chosen the system works as expected and the underlying adaptation is desirable or 'normal'. In such a case, we call the configuration state 'stable'. If a new configuration has a low probability we still assume that the system is working as usual and that this adaptation is due to a minor disturbance. Only when configurations with low probability are chosen over a longer course of time, we presume that a major disturbance or a system failure is on hand.

If it comes down to the implementation, configuration parameters are usually represented as real numbers. Since a single vector of real numbers has a probability of $P = 0$ in a continuous density, we will look at the probability densities that are created by several vectors.

By comparing the configurations of a subsystem in a current time window with those of a previous window, we can identify abnormal adaptation activity in a single subsystem. To identify abnormal changes in a SASO system at a global scale (i.e. at macro-level, we take all configuration comparisons into account and then apply a measure based on the "macroscopic measure for detecting abnormal changes in a multi-agent system" as defined by Kinoshita [28].

### 3.1. Definition of the Kinoshita measures

Kinoshita proposes a measure which he uses as an indicator for unusual activity changes in a distributed multi-agent system. Kinoshita defines two values, the activity factor at a given time and the variance of fluctuation of the activity factor. The activity factor is based on the classification of each agent as either active or inactive. Let $N$ denote the total number of agents in the multi-agent system and let $n_t$ be the number of active agents at a given time t. The activity factor $z_t$ is defined as

$$z_t := \frac{2 \cdot n_t - N + 1}{2 \cdot N} \tag{1}$$

For a given window size $M$, the fluctuation $\xi_t$ of the activity factor at time $t$ is calculated as

$$\xi_t := z_t - \frac{1}{M} \cdot \sum_{i=0}^{M-1} z_{t-i}$$

and finally, we calculate the variance $\nu_t$ of the fluctuation:

$$\nu_t := \frac{1}{M} \sum_{i=0}^{M-1} \xi_{t-i}^2 - \left( \frac{1}{M} \sum_{i=0}^{M-1} \xi_{t-i} \right)^2 \tag{2}$$

Kinoshita argues that unusual peaks in the time series of $\nu_t$ are an indicator of abnormal changes in the underlying distribution of active and inactive subsystems. Consequently, we aim at using this measure to detect such peaks that then serve as an indicator for undesired adaptation behaviour that needs to be suppressed.

### 3.2. Application of the Kinoshita measures

To apply the Kinoshita measure, we need to define what active and inactive agents are. Obviously, the subsystems of our SASO system are the agents in Kinoshita's measure. Our goal is to determine whether the configuration of each subsystem is stable or not. Therefore, we define a subsystem as active if and only if its configuration is unstable. That is that the divergence of the density of the latest configurations and the density of previous configurations is greater than a certain threshold. That means that for the latest configurations values were chosen that follow a distribution which significantly differs from the previous one.

Let $c_{a,t} \in \mathbb{R}^n$ be the $n$-dimensional configuration vector of the subsystem $a$ at time $t$. We model $c_{a,t}$ as a random variable. Let $M > 1$ be the window size and $L > 0$ be the delay parameter. Using an appropriate density estimation technique, we define two probability densities $DC_{a,t}$ and $DP_{a,t}$ based on parts of the time series of $c_{a,t}$:

$$DC_{a,t} := density(c_{a,t-m}, c_{a,t-m+1}, \ldots, c_{a,t})$$

$$DP_{a,t} := density(c_{a,t-l-m}, c_{a,t-l-m+1}, \ldots, c_{a,t-l})$$

$DC$ is the density of the current window and $DP$ the density of a previous window in the time series. To these two densities, we now apply the Kullback-Leibler divergence [29] measure

$$\begin{aligned} KL_2(P, Q) \quad &= \int_{-\infty}^{\infty} p(x) log\left(\frac{p(x)}{q(x)}\right) \Big/ 2 \\ &- \int_{-\infty}^{\infty} q(x) log\left(\frac{q(x)}{p(x)}\right) \Big/ 2 \end{aligned} \tag{3}$$

and get a new time series

$$d_{a,t} := KL_2(DC_{a,t}, DP_{a,t}) \tag{4}$$

Finally, we define a subsystem $a$ to be active at time $t$ if $d_{a,t} > \epsilon$ for a threshold value $\epsilon$. Otherwise, the subsystem is called inactive.

### 3.3. Finding the right parameters

Finding suitable values for the parameters $M$, $L$ and $\epsilon$ is crucial for the significance of the calculated Kinoshita measure. A low $\epsilon$ will mark configuration changes as unstable which at a closer look are still acceptable as stable. On the other hand, too high values for $M$ and $L$ would mask short term occurrences of high amplitude changes. Our experiments show that good choices for these parameters highly depend on the actual SASO-System. In general, the configuration of these parameters is consequently application-specific and should be customised at runtime using hyper-parameter-tuning. However, we aim at a 'good-enough' reference configuration.

A first approach to find such acceptable values is to create a simulation of the SASO system and gather the configuration time series of the simulated subsystems. In the next step, change these data samples for some of the agents such that at controlled time points for a few steps high amplitude changes occur. Then we create an optimisation problem: Find the best values for $M$, $L$ and $\epsilon$ such that the controlled changes create unique peaks in the Kinoshita measure.

Our experiments show that the best values vary depending on where we set the artificial changes. However, they vary only in a very limited range. Therefore, a reasonable next step is to run this optimisation several times with different points for the changes. This gives a distribution for $M$, $L$ and $\epsilon$. The values that occur most often usually are in fact good values for the actual evaluation.

Of course, this method only works if it is possible to create such artificial changes. If the base simulation already generates configuration samples with a wide range of values and a lot of changes in one time frame and very few changes in another, then it is hardly possible to find values that will give satisfying results. In this case, we consider the

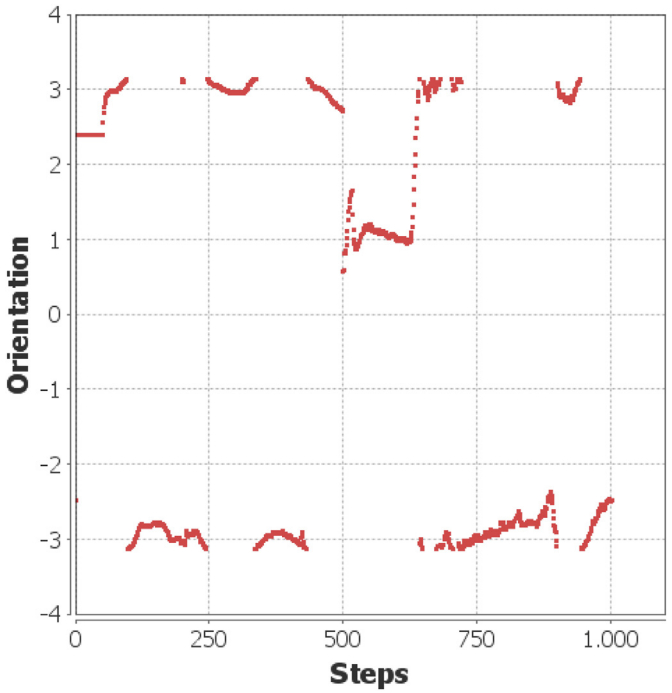


**Fig. 2.** The orientation time series of one of the affected birds. The values wrap around at $\pm\ \pi$. The disturbance is visible as the drop at $t = 500$.

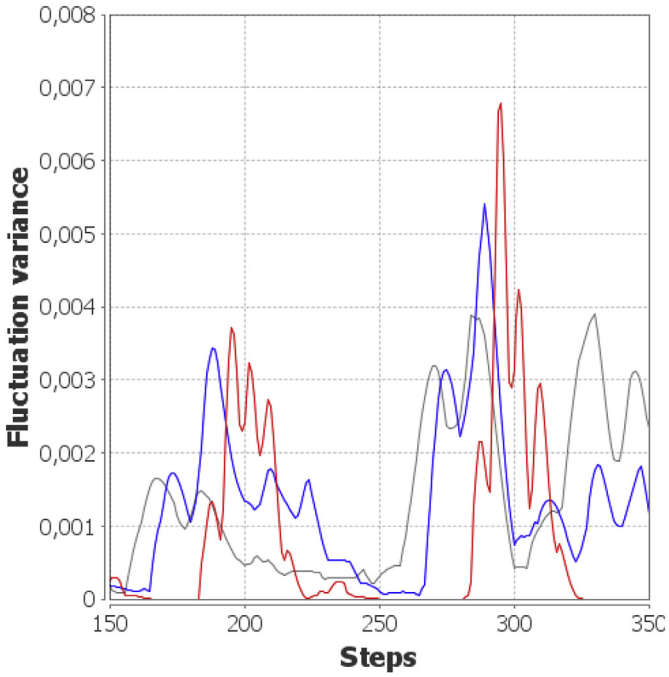


**Fig. 3.** Some results of the heuristic 3.3 for the first flocking simulation for a fixed $\epsilon = 1$. Red: $M = L = 10$, blue: $M = L = 20$ and grey: $M = L = 25$.

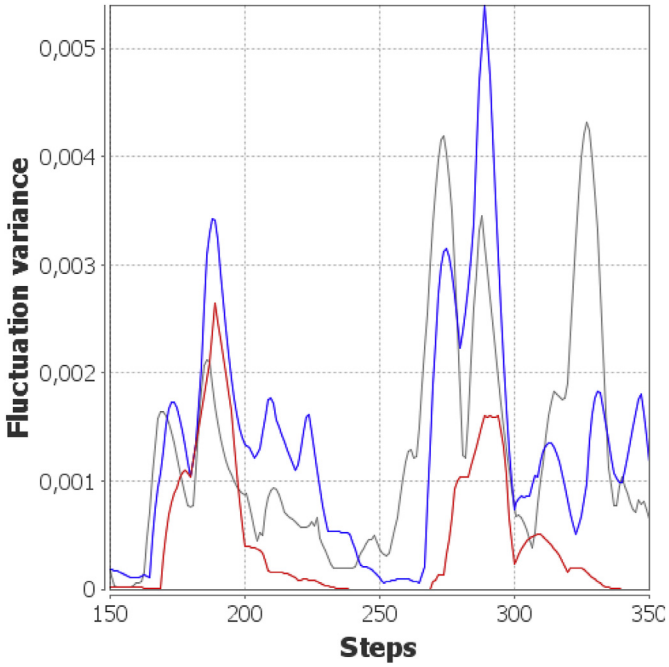


**Fig. 4.** Some results of the heuristic 3.3 for the first flocking simulation for a fixed $M = L = 20$. Red: $\epsilon = 2$, blue: $\epsilon = 1$ and grey: $\epsilon = 0.5$.

configuration of the system as unstable from the beginning. In fact, normal and abnormal adaptation behaviour cannot be distinguished in these scenarios, which renders our measure not applicable. However, current experiments show that this is not the case for the investigated scenarios (where future work aims also at defining the limits for the application).

## 4. Experiments and evaluation

We analyse the behaviour of the proposed measure in two artificial flocking [30] and two basic traffic simulations. All simulations are implemented in the MASON [31] simulation environment.

### 4.1. Flocks with external disturbance

This flocking simulation consists of 50 birds on a toroidal plane with random starting points and random initial orientations. We consider two cases: First a normal simulation and second a disturbed one. In both scenarios, the birds follow the usual rules of flocking:

- Alignment: A bird will align its direction with the average direction of its neighbours
- Cohesion: A bird will steer towards the centre of all neighbouring birds
- Avoidance: A bird will steer away from neighbours that are too close

For each of these rules, a direction vector $v_x$ is computed, weighted (with factor $w_x$) and then added together. This sum is then normalised and eventually added to the current direction $v_{current}$ vector of the given bird to determine it's direction $v_{next}$ in the next time step:

$$\begin{aligned} v_{next} &= v_{current} + normalise \\ &\left( w^{*}_{alignment} v_{alignment} \right. \\ &+ w^{*}_{cohesion} v_{cohesion} \\ &\left. + w^{*}_{avoidance} v_{avoidance} \right) \end{aligned} \tag{5}$$

For the two simulations, the values are $w_{cohesion} = w_{alignment} = 1.0$, $w_{avoidance} = 0.33$ and the sum is normalised to a value of 0.7.

In the undisturbed case, all birds follow these three rules. In the disturbed case, one bird $A_0$ is removed at time point $t = 500$. All birds within a distance of 50 units fly diametrically away from $A_0$ for two time steps and then follow their usual behaviour again (which simulates the response to a shot killing the bird, for instance).

The configuration on which we apply the measures is the orientation of the birds, represented as the angle against the x-axis (with values from $-\ \pi$ to $\pi$) of the simulation environment. Fig. 2 shows the time series of

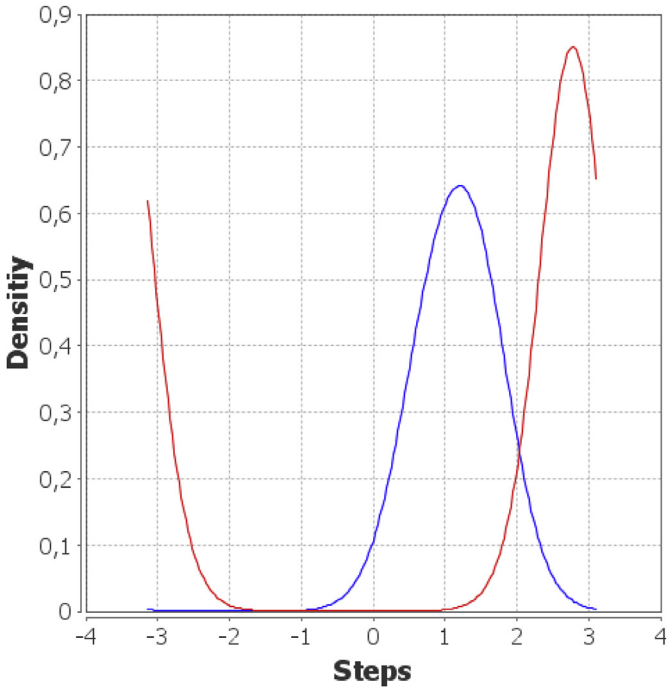


**Fig. 5.** Densities for the orientation distributions of bird from Fig. 2 in the disturbed flocking simulation. Red: during $t = 480$ to $t = 500$, blue: during $t = 500$ to $t = 520$.

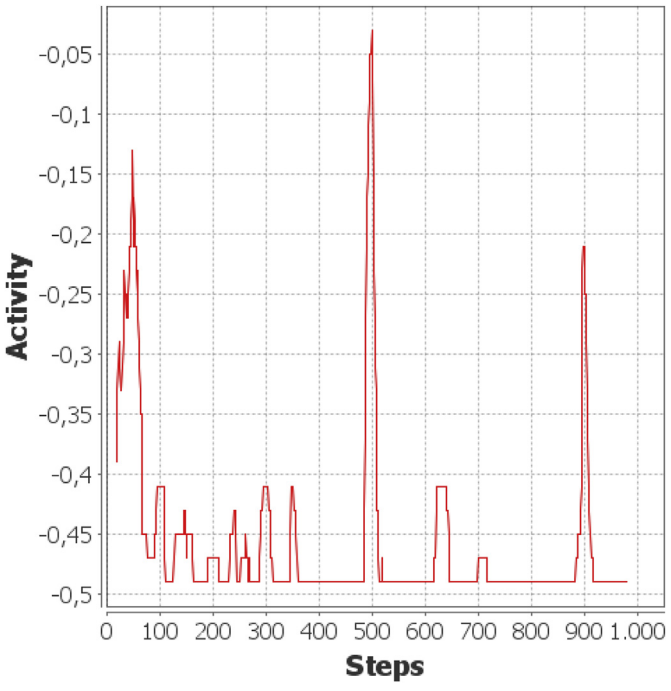


**Fig. 6.** The Kinoshita activity factor for the disturbed flocking simulation.

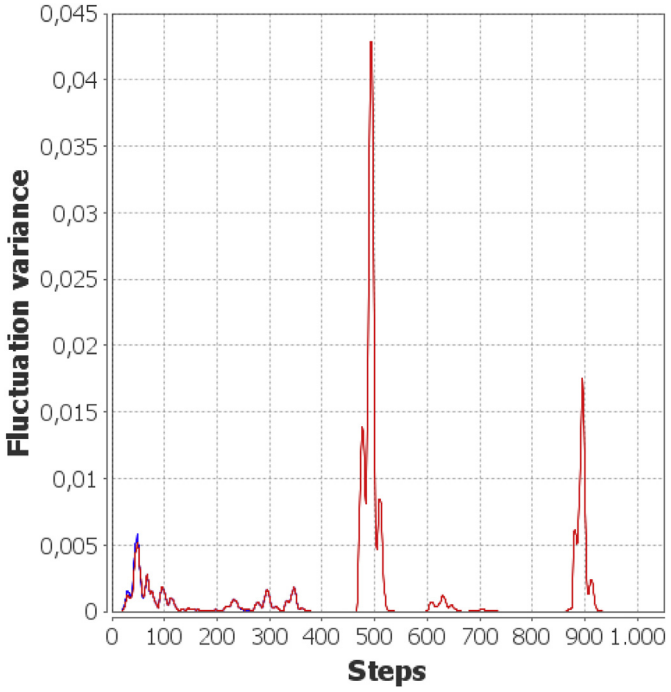


**Fig. 7.** The Kinoshita fluctuation variance of the flocking simulation. Red: the disturbed case, blue: undisturbed.

the orientation of one of the affected birds. Most of the time, this bird has an orientation near $\pm\,\pi$, but when the disturbance occurs it changes it's orientation abruptly to a value near 1. It then takes the bird about 120 steps to realign to the majority of the flock.

To apply the Kinoshita measure, we need to fix values for $M$, $L$ and $\epsilon$. For this, we put the undisturbed simulation into the heuristic from section 3.3 and change the configuration values for 10% of the agents. From $t = 200$ to $t = 210$ these agents get a constant configuration value of 1 and from $t = 300$ to $t = 310$ they get a value of 0. Therefore, we expect the heuristic to find values such that one peak at $t = 200 - L$ and one at $t = 300 - L$ is sufficiently prominent.

Fig. 3 shows results for a fixed $\epsilon = 1$. For $M = L = 10$ (red graph) we see the peaks at $t = 190$ and $t = 290$ but also two additional peaks at $t = 200$ and $t = 210$ which are almost the same height as the first one. $M = L = 25$ (grey graph) we have no sufficiently prominent peak at $t = 175$. The blue graph shows the result for $M = L = 20$ which meets our expectation fairly well.Fig. 4 gives an impression on the effects of changing the $\epsilon$-parameter. A low $\epsilon$ allows too many agents to be counted as active, even those who are not using artificially changed configuration data. Therefore, the grey graph ($\epsilon = 0.5$ gives too many peaks. The blue graph shows the results for $\epsilon = 1$, where we can identify the desired peaks alongside a few smaller ones. The red graph shows $\epsilon = 2$. Here, no other peaks are present. We then repeat this process with other time points for the artificially changed data. Finally, the heuristic tells us that good values for $M$ and $L$ are in fact near $M = L = 20$ and that $\epsilon \geq 1$ is suitable. We keep in mind that a high value for $\epsilon$ could suppress important agents. Therefore, we choose $M = L = 20$ and $\epsilon = 1$ for the evaluation of this simulation.

To determine whether the bird $a$ is count as active at $t = 520$ in the disturbed simulation, we need to calculate the distribution densities $DC_{a,520} = density(c_{a,500}, \ldots, c_{a,520})$ and $DP_{a,520} = density(c_{a,480}, \ldots, c_{a,500})$ of its orientation. To do so, we use the common kernel density estimation methods from Ref. [32]. Since the orientation is circular (wrapping around at $\pm\,\pi$), we need to use a circular kernel function. In this case, we chose a von Mises kernel [33]. Fig. 5 shows the resulting densities with the centre of distribution near 2.7 in the first time window and near 1.2 in the second window.

The $KL_2$-value for this bird is $d_{a,520} = KL_2\left(DC_{a,520}, DP_{a,520}\right) \approx 6.11$. This is higher than our threshold value. Consequently, the bird $a$ is considered to be 'active' at $t = 520$.

Doing this for all birds and all time points, we can calculate the Kinoshita activity factor (which is basically the ratio of active agents) using formula 1. The result is shown in Fig. 6. Here, we can already identify a peak at the disturbance event together with other phases of high activity. Finally, we can calculate the fluctuation variance $\nu_t$. Fig. 7 shows the resulting time series for the disturbed (red) and the

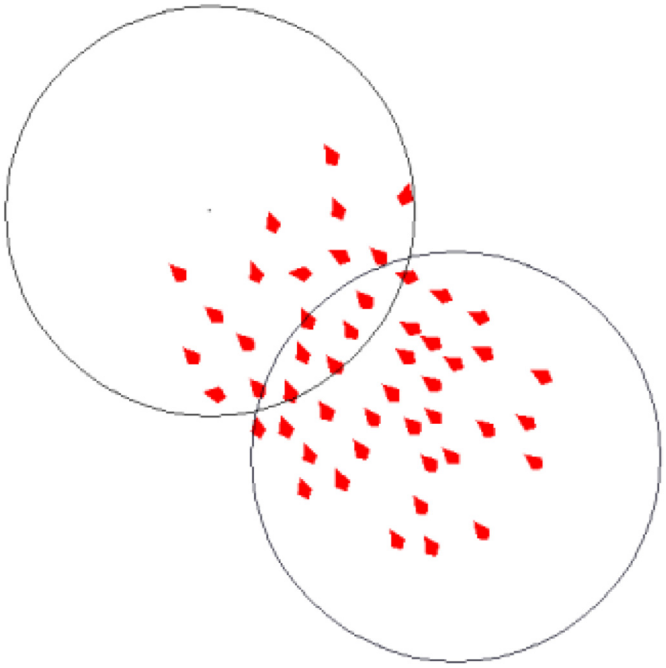

**Fig. 8.** The second flocking simulation at time step $t = 250$. The agents are heading towards the upper left area which has a higher utility value than the lower right area since $t = 205$.

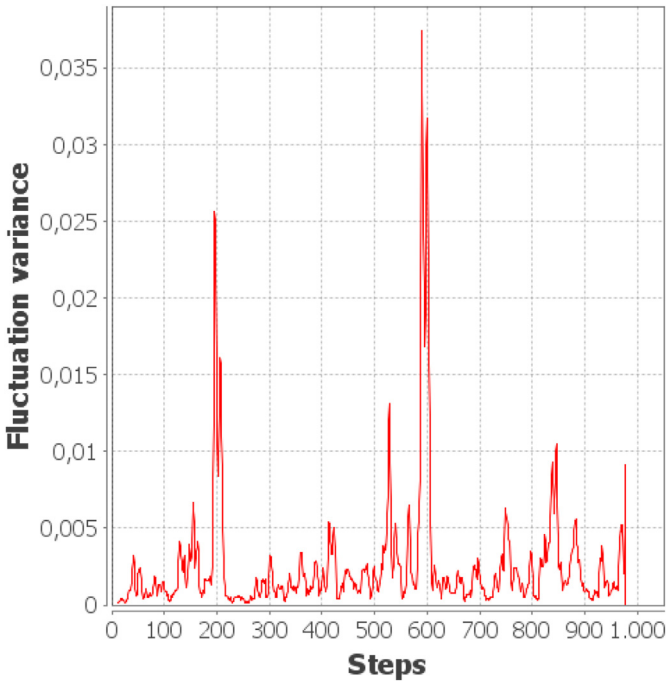


**Fig. 9.** The Kinoshita fluctuation variance for the utility changing simulation. The peaks at $t = 205$ and $t = 588$ show the reaction of the agents to the change of the highest utility area.

undisturbed (blue) simulation. Except for minimal differences due to the arithmetic precision flaws of the Java double type [34] both graphs are identical until $t = 480$. At that point, the time windows incorporate the disturbance. The disturbance creates a clear peak in the graph. Shortly after the disturbance, the birds group in three isolated flocks, two of them merge at $t = 900$ which results in a second peak. Other than the activity factor, the fluctuation variance emphasises these two events.

### 4.2. Flocks with change of utility

In this simulation, we model 50 agents that again roam a toroidal plane with random starting points and random initial orientations. Next, we assign each point of the plane a utility value such that the utility values for all points in the plane are zero except for two circular, intersecting areas. The agents follow the rule of avoidance as in the previous scenarios (but not alignment or cohesion, i.e. $w_{cohesion} = w_{alignment} = 0$) and they can evaluate the utility value of their current position. The goal for each agent is to stay inside the area of the highest utility value. If it is not inside that area it will steer towards the centre of that area until the area is entered (see Fig. 8). This can be viewed as a flock of predators running around in an area with the best supply of prey or as a group of vacuum cleaning robots that clean the dirtiest area in the house.

In analogy to the real world, where the predators will decimate the prey and the robots will remove the dirt, we introduce a dynamic in the utility values. The values $U$ for the two non-zero utility areas change over time following a sinus function with a constant phase $p$ and the time step variable $t$:

$$U(t) := (sin(0.001^{*} p^{*} t) + 1)/2$$

The phase parameters for the two areas are $p = 7$ and $p = 9$. Therefore, their utility values oscillate between 0 and 1 with different speeds. By this means, we force the agents to migrate to the other area which creates a notable overlay over the usual chaotic motion behaviour. At the beginning of the simulation, the lower right area is the one with the highest utility value. Within 150 steps, all agent have reached this area. At $t = 205$ the upper left area will become the one with the higher value

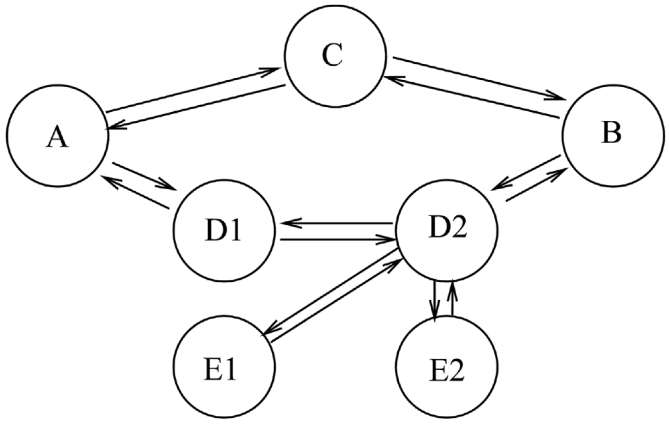


**Fig. 10.** The street network for traffic simulation. The circles are intersections and the arrows are the street lanes.

and the agents start migrating towards it. At $t = 588$ the roles will switch again.

Putting the time series of the agents' orientation into our measures and applying the parameters $M = L = 12$ and $\epsilon = 0.8$ as given by the heuristic creates the graph for $\nu_t$ as shown in Fig. 9. The two prominent peaks show the reaction of the agents to the changing utility. They are an indicator for a major configuration change that a system user should examine.

### 4.3. Network of traffic lights with internal disturbance

The third scenario is inspired by the Organic Traffic Control (OTC) system that self-adapts the green duration of traffic lights [35], establishes progressive signal systems [36], and guides drivers through the network using variable message signs [37]. Here, we used an abstract traffic simulation, where the SASO system consists of seven interconnected intersection controllers. Fig. 10 shows the layout of the street

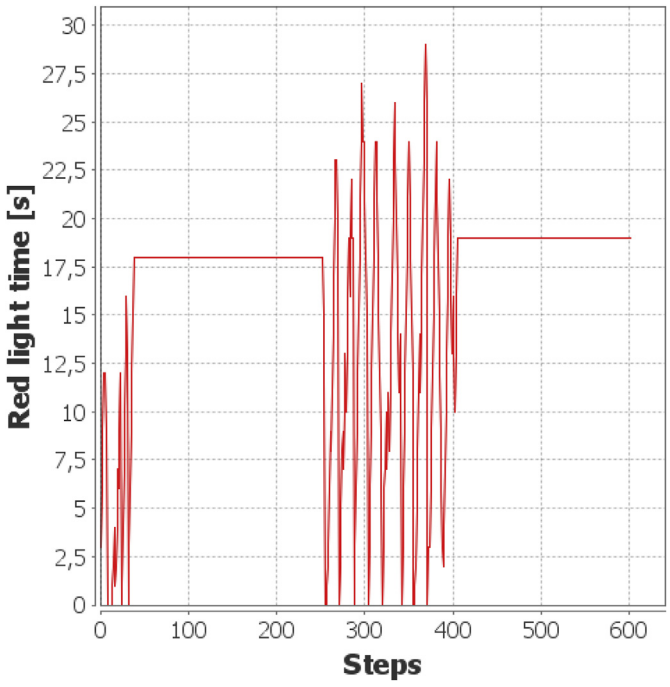


**Fig. 11.** The red light times at intersection $D2$ for the lane $E2 \rightarrow D2$.

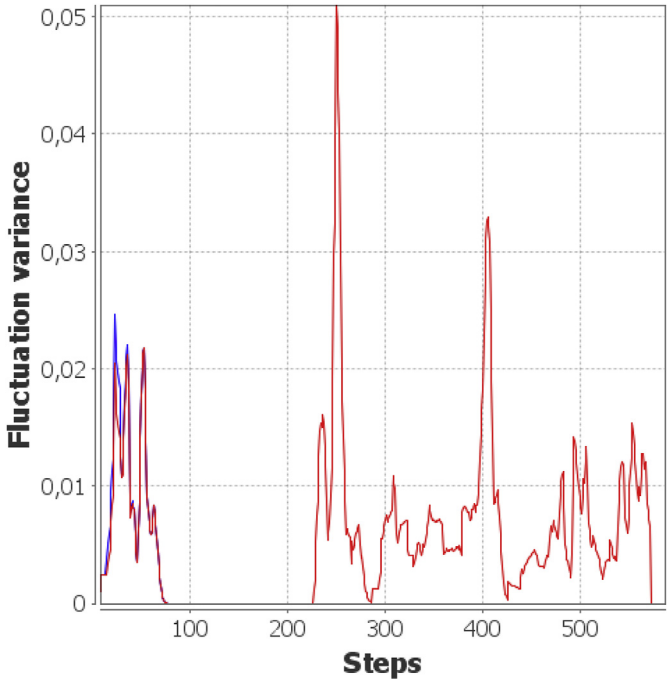


**Fig. 12.** The Kinoshita fluctuation variance of the first traffic simulation. Red: with a blocked intersection from $t = 250$ to $t = 400$, blue: undisturbed.

network.

Each intersection tries to minimise the waiting time for all cars at all incoming lanes by optimising the red light times for each lane. There are 250 cars in the simulation. They are part of the environment and not of the SASO system itself. Each car picks its destination randomly and chooses the shortest path to it. When it reaches the destination, the process is repeated and therefore all cars in motion constantly. In contrast to OTC system, the intersections of the SASO system in this simulation do not communicate with each other. Their reconfiguration results are only

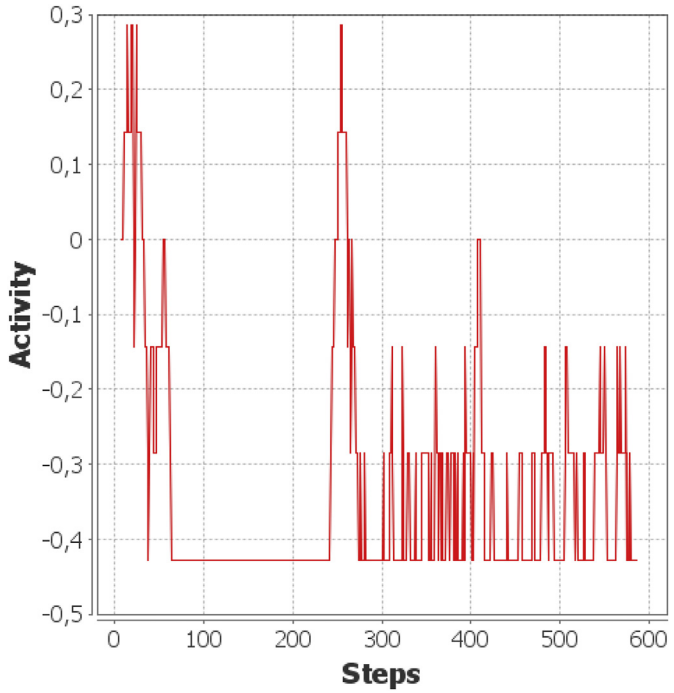


**Fig. 13.** The Kinoshita activity factor of the first traffic simulation with a blocked intersection from $t = 250$ to $t = 400$.

based on the current traffic volume at intersection's lanes.

The optimisation function to be minimised is the sum of all waiting times of all cars in all lanes. We assume that each car needs a constant time to cross the intersection if it is in the first position of an incoming lane with a green light. Therefore, the waiting time for one car is defined by the number of cars that are processed before the current car leaves the intersection. In one time step of the simulation, several cars can be processed. The waiting time of the unprocessed cars is carried over to the next simulation step. After each simulation step, the intersections look at the waiting times of the unprocessed cars and try to find a red light configuration that minimises the expected waiting time over all cars in the next step.

In this scenario, the path between intersections $A$ and $B$ is chosen with a probability of $p = 0.9$ and the path between $E1$ and $E2$ is chosen with $p = 0.75$. Consequently, these two separate paths have the highest traffic volume. We then block the central intersection $C$ between $t = 250$ and $t = 400$. During this time, the intersection $C$ is unavailable and the cars have to use other paths. Therefore, the intersection $D2$ will have to reconfigure its red-light times to handle both traffic streams. Fig. 11 shows the results of the disturbance for one lane at the intersection $D2$.

We re-used the heuristic approach from the previous scenarios, which chose a window size of $M = 20$ steps, a delay of $L = 20$ and a threshold $\epsilon = 10$ to generate the time series $\nu_t$. Fig. 12 shows $\nu_t$ for the simulation with (red) and without (blue) the disturbance. After an initial setup phase, which takes 70 steps, there is no longer a difference between currently and previously chosen configurations in the undisturbed case. The configuration is stable for all $t > 70$. In the disturbed case, on the other hand, the instability of the configuration is clearly visible. The start and the end of the blockade require high reconfiguration effort of the intersections to handle the changed traffic flow. The effects manifest themselves as the two peaks at $t = 250$ and $t = 400$. In the beginning of the simulation, and during and after the blockade, there are still configuration changes that are big enough to count as active but they generate no isolated peaks. For the sake of completeness, We provide the Kinoshita activity factor for this simulation in Fig. 13. In this time series, the signals for the blockade are not as prominent and the setup phase creates similar signals.

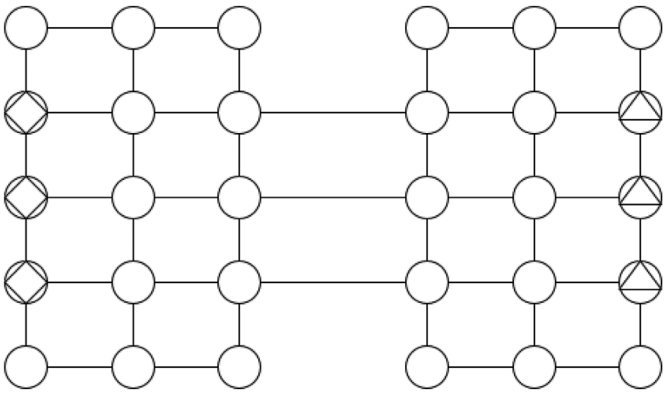

**Fig. 14.** The street network for the second traffic simulations. Home steads are marked with a square, work places with a triangle.

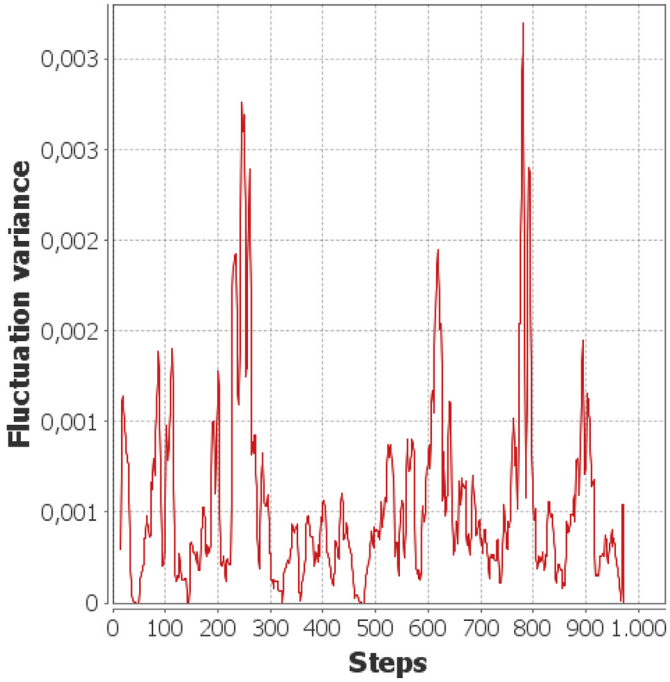


**Fig. 15.** The Kinoshita fluctuation variance of the rush hour simulation with increased traffic flow starting at $t = 250$ and $t = 750$.

### 4.4. Network of traffic lights with changes in the environment

We created this additional simulation to test the measures in a more noisy and larger environment where a smaller part of the subsystems experience a direct influence from external events. The scenario simulates rush hours in a city street network. The traffic lights and the cars operate in the same way as in the previous scenario, but we use a different street network. The network represents two islands each with a Manhattan-type network of sizes 3 by 5. The islands are connected with three bridges. The connections between two intersections provide one lane for each direction (see Fig. 14).

During the whole simulation, there are 250 cars driving around randomly. Furthermore, three intersections on one island are designated to be homesteads and three intersections on the other island are labelled as work places. At $t = 250$ a total count of 500 new cars will appear randomly in the three homesteads and start driving towards one of the three workplaces. Therefore, the bridges have to handle an increasing flow from one side to the other in the following time steps. When the new cars arrive at their work places, they are removed from the simulation. At $t = 750$ this is repeated but this time the new cars will go from the workplaces towards home.

Fig. 15 shows the time series $\nu_t$ for this scenario with parameters $M = L = 15$ and $\epsilon = 2$. Since the random background traffic has no preferred routes, we see a more noisy time series with more peaks. The peaks occurring after $t = 250$ and $t = 750$ are due to the increased configuration changes that are caused by the higher traffic flow. These peaks are still clearly visible but are not as outstanding as in the previous simulations. The other peaks are a result of the random traffic. This shows that the measure is still applicable in such an environment although its performance to identify major events is not as good as in situations with less noise.

## 5. Conclusion

This article presented a step towards an integrated framework for the quantification of runtime SASO behaviour. Based on preliminary work on a probabilistic approach to assess a degree of self-adaption and self-organisation, this article presented a method to measure the stability of configurations of autonomous subsystems within an overall SASO constellation. Taking the design concepts of Organic Computing and Autonomic Computing as basis, we found our approach on the interface between adaption mechanism and productive system by assuming read access to the current configurations of the productive units. We showed that the proposed measure building upon the Kinoshita approach can be used to identify global anomalies in the system's configuration which serve as indicators for abnormal adaptation processes. The capability to identify such anomalies relies on the choices for the incorporated hyper-parameters.

The presented approach is a step towards the overall goal of developing an integrated measurement framework for a continuous runtime analysis of SASO system properties. Future work follows two major research directions: On the one hand, we aim at an application to further, more complex scenarios of distributed SASO constellations and assess the behaviour of the measure. This includes an evaluation of this method in real-world applications and finding more constructive methods to identify suitable parameters. On the other hand, we build on-top of the indicators provided by the measure to establish a runtime approach to using this information within the adaption logic, e.g., to better balance the trade-off between possible performance gain due to adaptation decisions and the impact on the stability and correspondingly the users' acceptance.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgements

This research is partly funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under grant TO 843/5-1 (project 'InTURN'). The publication of this article was financially supported by DFG within the funding programme Open Access Publizieren. The authors acknowledge the support by the DFG.

## References

[1] Bellman K, Botev J, Diaconescu A, Esterle L, Gruhl C, Landauer C, Lewis PR, Nelson PR, Pournaras E, Stein A, Tomforde S. Self-improving system integration: mastering continuous change. Future Generat Comput Syst 2021;117:29–46. https://doi.org/10.1016/j.future.2020.11.019. http://www.sciencedirect.com/science/article/pii/S0167739X20330430.

[2] Calma A, Kottke D, Sick B, Tomforde S. Learning to learn: dynamic runtime exploitation of various knowledge sources and machine learning paradigms. In: 2017 IEEE 2nd international workshops on foundations and applications of self* systems; 2017. p. 109–16.

[3] Rajkumar R, Lee I, Sha L, Stankovic J. Cyber-physical systems: the next computing revolution. In: Design automation conference. IEEE; 2010. p. 731–6.
[4] Atzori L, Iera A, Morabito G. The internet of things: a survey. Comput Network 2010;54(15):2787–805.
[5] Kounev S, Lewis P, Bellman K, Bencomo N, Camara J, Diaconescu A, Esterle L, Geihs K, Giese H, et al. The notion of self-aware computing. In: Self-aware computing systems. Springer; 2017. p. 3–16.
[6] Bellman K, Tomforde S, Würtz R. Interwoven systems: self-improving systems integration. In: 8th IEEE int. Conf. on self-adaptive and self-organizing systems workshops; 2014. p. 123–7.
[7] Kephart J, Chess D. The vision of autonomic computing. IEEE Computer 2003; 36(1):41–50.
[8] Müller-Schloer C, Tomforde S. Organic computing – techncial systems for survival in the real world, autonomic systems. Birkhäuser Verlag; 2017.
[9] Weyns D, Schmerl B, Grassi V, Malek S, Mirandola R, Prehofer C, Wuttke J, Andersson J, Giese H, Göschka K. On patterns for decentralized control in self-adaptive systems. In: Software engineering for self-adaptive systems II. Springer; 2013. p. 76–107.
[10] D'Angelo M, Gerasimou S, Ghahremani S, Grohmann J, Nunes I, Pournaras E, Tomforde S. On learning in collective self-adaptive systems: state of practice and a 3d framework. In: 2019 IEEE/ACM 14th international symposium on software engineering for adaptive and self-managing systems. IEEE; 2019. p. 13–24.
[11] Krupitzer C, Roth F, VanSyckel S, Schiele G, Becker C. A survey on engineering approaches for self-adaptive systems. Pervasive Mob Comput 2015;17:184–206.
[12] Tomforde S, Kantert J, Müller-Schloer C, Bödelt S, Sick B. Comparing the effects of disturbances in self-adaptive systems - a generalised approach for the quantification of robustness. Trans. Comput. Collect. Intell. 2018;28:193–220.
[13] Tomforde S, Goller M. To adapt or not to adapt: a quantification technique for measuring an expected degree of self-adaptation. Computers 2020;9(1):21.
[14] M. Goller, S.Tomforde, Towards a continuous assessment of stability in (Self-) Adaptation behaviour, Proceedings of the 1st IEEE international conference on autonomic computing and self-organising systems workshops (ACSOS-W), 6th Workshop on Self-Aware Computing (SeAC20).
[15] Tomforde S, Prothmann H, Branke J, Hähner J, Mnif M, Müller-Schloer C, Richter U, Schmeck H. Observation and control of organic systems. In: Müller-Schloer C, Schmeck H, Ungerer T, editors. Organic computing - a paradigm shift for complex systems, autonomic systems. Birkhäuser Verlag; 2011. p. 325–38.
[16] Tomforde S, Sick B, Müller-Schloer C. Organic computing in the spotlight, CoRR abs/1701.08125. http://arxiv.org/abs/1701.08125.
[17] Tomforde S, Kantert J, Sick B. Measuring self-organisation at runtime - a quantification method based on divergence measures. In: Proc. of 9th int. Conf. on agents and artificial intelligence; 2017. p. 96–106.
[18] Fisch D, Jänicke M, Sick B, Müller-Schloer C. Quantitative emergence–a refined approach based on divergence measures. In: IEEE int. Conf. on self-adaptive and self-organizing systems. IEEE; 2010. p. 94–103.
[19] Chan W. Interaction metric of emergent behaviours in agent simulations. In: Proc. of the winter sim. Conf.; 2011. p. 357–68.
[20] Raibulet C, Masciadri L. Metrics for the evaluation of adaptivity aspects in software systems. Int. J. Adv. Softw. 2010;3(1 & 2):238–51.
[21] Kaddoum E, Raibulet C, Georgé J-P, Picard G, Gleizes M-P. Criteria for the evaluation of self-* systems. In: Pro. of ICSE works. on softw. Eng. for adaptive and self-managing sys.; 2010. p. 29–38.
[22] Cámara J, Correia P, de Lemos R, Vieira M. Empirical resilience evaluation of an architecture-based self-adaptive software system. In: Pro. of 10th int. ACM sigsoft conf. on quality of softw. Architectures; 2014. p. 63–72.
[23] Gronau N. Determinants of an appropriate degree of autonomy in a cyber-physical production system. In: Proc. of 6th Int. Conf. on Changeable, Agile, Reconfigurable, and Virtual Production, 52; 2016. p. 1–5.
[24] Eberhardinger B, Anders G, Seebach H, Siefert F, Reif W. A research overview and evaluation of performance metrics for self-organization algorithms. In: Proc. of self-adaptive and self-organizing systems works. SASO-W'15); 2015. p. 122–7.
[25] Filieri A, Maggio M, Angelopoulos K, D'ippolito N, Gerostathopoulos I, Hempel A, Hoffmann H, Jamshidi P, Kalyvianaki E, Klein C, Krikava F, Misailovic S, Papadopoulos A, Ray S, Sharifloo A, Shevtsov S, Ujma M, Vogel T. Control strategies for self-adaptive software systems. ACM Trans Autonom Adapt Syst 2017;11(4):24: 1–24:31.
[26] Chen T, Bahsoon R, Wang S, Yao X. To adapt or not to adapt? technical debt and learning driven self-adaptation for managing runtime performance. In: Proc. of the 2018 ACM/SPEC int. Conf. on Performance Engineering; 2018. p. 48–55.
[27] Tomforde S. From "normal" to "abnormal": a concept for determining expected self-adaptation behaviour. In: IEEE 4th international workshops on foundations and applications of self* systems; 2019. p. 126–9.
[28] Kinoshita T. Basic characteristics of a macroscopic measure for detecting abnormal changes in a multiagent system. MDPI Sensors 2015;15(4):9112–35.
[29] Bishop C. Pattern recognition and machine learning. second ed. Information Science and Statistics, Springer; 2011.
[30] Reynolds CW. Flocks, herds and schools: a distributed behavioral model. ACM SIGGRAPH Comput. Graph. 1987;21:25—34.
[31] Luke S, Cioffi-Revilla C, Panait L, Sullivan K, Balan G, Mason. A multi-agent simulation environment. Trans. Soc. Model. Simul. Int. 2005;82(7):517–27.
[32] Parzen E. On estimation of a probability density function and mode. Ann Math Stat 1962;33:1065—1076.
[33] Mardia K, Jupp P. Directional statistics. Wiley; 1999.
[34] D. Goldberg, What every computer scientist should know about floating point arithmetic, ACM Comput Surv 23. doi:10.1145/103162.103163.
[35] Prothmann H, Rochner F, Tomforde S, Branke J, Müller-Schloer C, Schmeck H. Organic control of traffic lights. In: Autonomic and trusted computing, 5th international conference, ATC 2008, Oslo, Norway, june 23-25, 2008, proceedings; 2008. p. 219–33.
[36] Tomforde S, Prothmann H, Rochner F, Branke J, Hähner J, Müller-Schloer C, Schmeck H. Decentralised progressive signal systems for organic traffic control. In: Second IEEE international conference on self-adaptive and self-organizing systems, SASO 2008, 20-24 october 2008, Venice, Italy; 2008. p. 413–22.
[37] Prothmann H, Schmeck H, Tomforde S, Lyda J, Hähner J, Müller-Schloer C, Branke J. Decentralised route guidance in organic traffic control. In: 5th IEEE international conference on self-adaptive and self-organizing systems, SASO 2011, ann arbor, MI, USA, october 3-7, 2011; 2011. p. 219–20.