The definite version is published in Proc. IEEE ACSOS, Washington, DC, US, 2021.

Swarmalators with Stochastic Coupling and Memory

Udo Schilcher, Jorge F. Schmidt, Arke Vogell, and Christian Bettstetter
University of Klagenfurt, Institute of Networked and Embedded Systems, Austria
email: udo.schilcher@aau.at

***Abstract*—Swarmalators combine two types of distributed coordination — swarming and synchronization — whose mutual coupling leads to the emergence of spatio-temporal patterns. This paper studies issues for implementing swarmalators in the real world. First, the model is discretized in time to achieve limited communication overhead. We propose to apply *stochastic coupling*, in which entities broadcast their states at reduced rate, and introduce *memory* in each entity to store recently received state updates. The resulting system still converges to the original patterns. We investigate the convergence time and provide a lower limit for the rate of state exchange to ensure reasonable performance. Second, we show that inaccurate localization and physical size often have no impact on the pattern emergence, whereas limits for speed and acceleration lead to a slowdown. Our work is from the perspective of computing and robotics, but the approach and results can potentially be applied to other fields as well.**

***Index Terms*—Synchronization, swarming, stochastic coupling, emergence, self-organizing systems**

1. Introduction

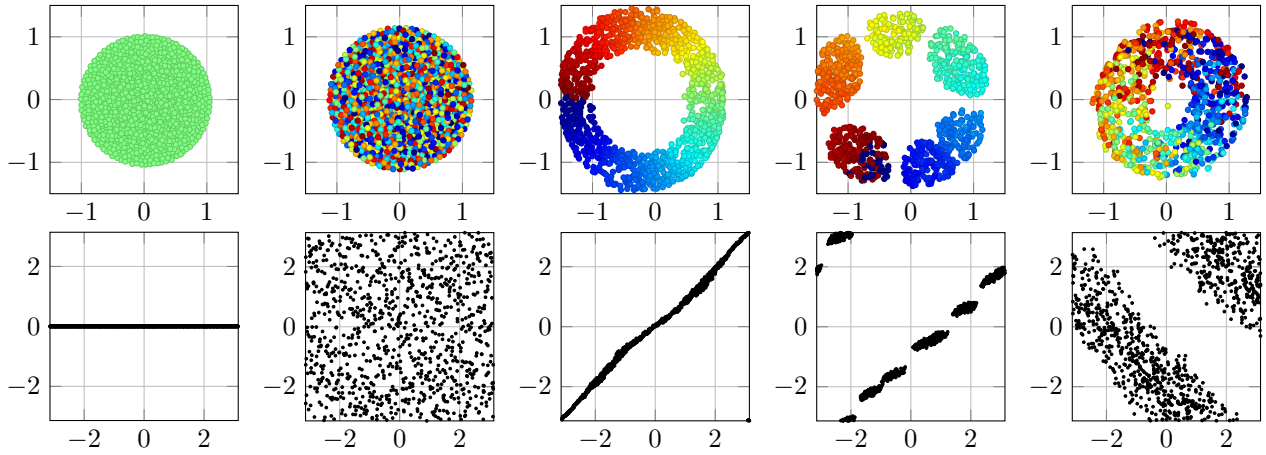
Nature is a source of inspiration for engineers to find sweet spots between robust and efficient design. In autonomous multi-agent systems, two well-known examples of nature-inspired solutions are swarming and synchronization [1]. Swarming typically coordinates a group of entities (e.g., robots or drones [2], [3], [4], [5]) to fulfill a given task. Its origins lie in biology, where groups of animals stay close together to move as a unit (e.g., as a bird flock [6]). A swarm emerges as each of the entities follows some simple rules and interacts with its local neighbors. Also synchronization is observed in nature (e.g., in fireflies and heart cells that blink and beat in unison [7]). Again, the behavior emerges by obeying simple rules and local interactions. It can be modeled by pulse coupled oscillators [8], [9], which allows a transfer to technical solutions [10]. Both swarming and synchronization are self-organizing processes. There is no single point of failure; simple rules imply that powerful hardware is not required; and the distributed nature shows robustness against system dynamics, e.g., entities joining and leaving the system.

Swarming and synchronization were modeled in different ways but have recently been merged into one integrated model: the swarmalator [11]. It allows to simultaneously form a swarm and synchronize an internal phase. The changes of location and phase of any swarmalator are each impacted by both the distances to and phases of neighboring swarmalators. This interplay between phases and locations yields novel system dynamics not observed before. In particular, new stable states of the swarming occur, which are possible only due to the stabilizing effect of the phases [12].

To apply the swarmalator model to technical systems, several constraints need to be considered (some of which have been highlighted in [13]): communication is limited in terms of throughput, latency, and information loss; localization has limited accuracy; and movement is constrained by physics and hardware capabilities. These constraints could lead to a different outcome than in the original model, which is subject to our investigation. Particularly, we are interested in the impact of technical constraints on the dynamics of swarmalators. Our contributions are:

- We employ the concept of **stochastic coupling** [14] to randomize the interactions between swarmalators and introduce a **memory** in each swarmalator to store the most recently received states.
- We show that this extension leads to a reduction of up to $95\,\%$ in communication overhead without relevant changes in the resulting patterns and the convergence time.
- We analyze the effects of system parameters, such as temporal resolution and memory initialization.
- We investigate the effects of a limited communication range and its controlled variation over time on the emerging patterns and show that both pattern and convergence time depend on the range.
- We extend the original swarmalator model to account for constraints imposed by the hardware, e.g., analyzing the effects of locomotion and localization.
- We provide insights on the pattern formation and analyze the dynamics, convergence time, and resource consumption based on the modified model and compare it to the idealized original model.

The rest of the paper is organized as follows: Section 2 specifies the mathematical model. Section 3 applies the concept of stochastic coupling and analyzes its impact on swarmalators. Section 4 addresses the implications of limiting speed and acceleration and of introducing a swarmalator size on the emerging patterns. Finally, Section 5 concludes.



(a) Static sync (b) Static async (c) Static phase wave (d) Splintered phase wave (e) Active phase wave

Figure 1. Converged patterns for swarmalators with stochastic coupling and memory initialized with 0 for coupling probability $p = 0.01$, parameters J and K according to Table 1, a simulation time of 100 (=1000 iterations for $\Delta t = 0.1$) for the static cases, 200 for *active phase wave*, and 1000 for *splintered phase wave*, with $N = 1000$. Upper row: swarmalator locations (x and y coordinates) and phases (colors from blue to red). Lower row: swarmalator phases (y-axis) over polar angle of their locations (x-axis).

2. Original Swarmalator Model

A swarmalator i has location x_i and internal phase θ_i. The behavior of N swarmalators can be controlled by two differential equations [11]:

$$\dot{x}_i = \frac{1}{N}\sum_{j=1}^{N} \frac{x_j - x_i}{\|x_j - x_i\|}(1 + J\cos(\theta_j - \theta_i)) - \frac{x_j - x_i}{\|x_j - x_i\|^2}$$
$$\dot{\theta}_i = \frac{K}{N}\sum_{j=1}^{N} \frac{\sin(\theta_j - \theta_i)}{\|x_j - x_i\|} \qquad (1)$$

for $i = 1, \ldots, N$ with two parameters J and K. The first equation controls how a swarmalator changes its *position* depending on the relative positions and phases of the other swarmalators. It combines two forces: The first force attracts swarmalators with similar phases and repels swarmalators with different phases if $J > 0$, and vice versa if $J < 0$. The second force leads to stronger repulsion when swarmalators are very close to each other in order to avoid collisions. The second equation controls how a swarmalator changes its *phase* depending on the positions and phases of the other swarmalators, which resembles the Kuramoto model [8]. Swarmalators tend to synchronize their phases if $K > 0$; they tend to desynchronize in the sense that neighboring swarmalators aim to maximize their phase differences if $K < 0$; and no phase adjustments take place if $K = 0$. Depending on J and K (see Table 1), the system converges to one of five patterns [11] shown in Fig. 1.

In order to find a particular solution of (1), we need to provide initial conditions. In this work, we use random initial conditions: At time $t = 0$, the locations x_i are sampled i.i.d. from a uniform distribution with support $[-1, 1] \times [-1, 1]$, and the phases θ_i are sampled i.i.d. uniformly from $[-\pi, \pi]$.

TABLE 1. SPACE-TIME PATTERNS WITH PARAMETER VALUES OF [11].

Pattern	J	K
Static sync	0.1	1
Static async	0.1	-1
Static phase wave	1	0
Splintered phase wave	1	-0.1
Active phase wave	1	-0.75

The swarmalator model can be applied in computing and engineering [15], where its ability to control multiple robots and drones has been demonstrated [13]. In such implementations, the swarmalators need to exchange their locations and phases by means of messages [16]. Hence, they cannot implement the continuous coupling used in the original model but require a time-discrete coupling. Such discretization is implicitly considered in the simulations of the original model, as they are based on the Euler method to numerically solve the differential equations [11]. Euler applies an iterative process to discretize time: in each step t, the values of the derivatives $\dot{x}$ and $\dot{\theta}$ are calculated and then added to the current state: $x(t+1) = x(t) + \dot{x}(t)\Delta t$ and $\theta(t+1) = \theta(t) + \dot{\theta}(t)\Delta t$, where Δt is the step size. It is important to choose a suitable Δt-value that is small enough to obtain convergence to a solution but also large enough to avoid numeric inaccuracies, which can add up to large deviations from the continuous solution [17].

3. Stochastic Coupling with Memory

A challenge in using the swarmalator model in practice is to limit the communication overhead for exchanging states. The model needs to be adapted and tuned to implement this restriction. We investigate potential changes of system dynamics caused by such modifications. Interesting questions are: Do the same patterns emerge? If so, how long

TABLE 2. SIMULATION PARAMETERS

Parameter	Symbol	Value
Initial area size		2×2
Swarmalator count	N	100
Max number of steps		10 000
Step size	Δt	0.1
Pattern		Static sync
Memory initialization		Zeroes
Stochastic coupling	p	0.1
Maximum communication range	d^*	∞
Localization noise	σ	0
Speed threshold	v^*	0.001
Minimum gap size	g^*	0.15
Phase spreads threshold	s^*	3.5
Bandwidth threshold	b^*	2.5
Swarmalator size	s	0

do they take to converge? In the following simulations, the parameters of Table 2 are used unless other values are given.

3.1. Motivation and Concept

Instead of exchanging the system state in every Euler step calculation, we revert to a probabilistic message exchange: in each step, a swarmalator shares its location and phase with probability $p \leq 1$ only. This concept of "stochastic interactions" has already been suggested for pure synchronization (without swarming) for convergence reasons [14]. In the context of swarmalators, we now do it for two reasons:

- The states of all swarmalators have to be transmitted every Δt time units. Hence, the traffic load on the wireless link can become high (depending on Δt, the number of swarmalators, and the means of communication). This signaling traffic consumes parts of the bandwidth needed by other applications as well. This applies e.g. to long-range communication setups limited by low data rates [18].
- Some messages are lost due to the physical nature of the wireless channel. Even under good channel conditions not all messages are correctly received by all swarmalators. The message exchange over wireless channels has a probabilistic character anyway.

For these reasons, we propose the following: Each swarmalator maintains a small memory that stores the state information received from the other swarmalators. This information is updated whenever a new state update is received. Every time an Euler step has to be calculated, the system state is read from the memory.

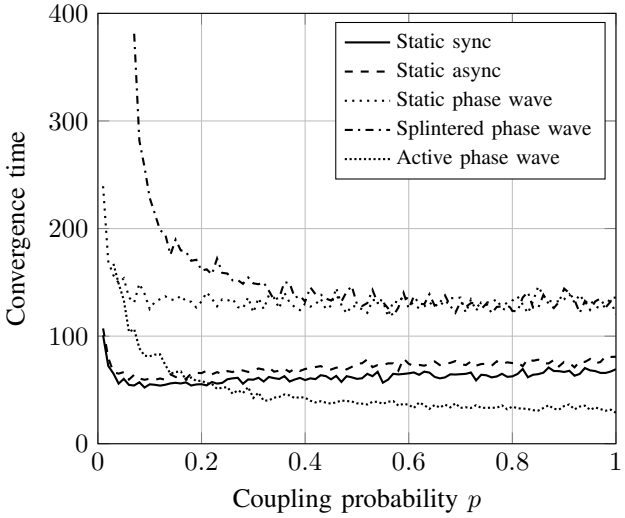
3.2. Patterns and Convergence Time

We perform a series of simulations to address the question as to whether or not the five patterns of the original swarmalator model emerge if we include stochastic coupling and memory. We simulate $N = 1000$ swarmalators that are coupled using a probability $p = 0.01$ over 100 time units. Fig. 1 shows the resulting patterns in space and the corresponding phases; Table 1 gives the values for J and K. The phases are plotted over the azimuth angle of the locations, which illustrates the spatial ordering of the phases. We find that the same five patterns emerge under the given parameters in both the spatial and the phase domains.

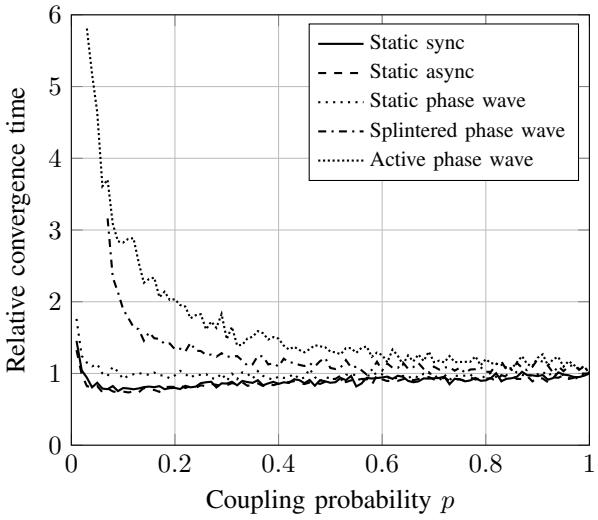
To analyze the impact of p on the convergence speed, we simulate (1) with variable duration where the simulation stops when convergence occurs. The convergence criterion depends on the pattern:

- *Static patterns:* In all three static patterns (sync, async, and phase wave), we observe the following: At the beginning, the swarmalators move around and adjust their phases quickly, i.e., $|\dot{x}_i|$ and $|\dot{\theta}_i|$ are high. Over time, these values tend to decrease and approach zero for $t \to \infty$, which leads to the resulting patterns. We say that the system has converged if $\max_i |\dot{x}_i| < v^*$, where $v^* > 0$ is some threshold for the speed of the fastest swarmalator; it should be low, at least below the speed of the initial movements. We do not apply such a bound to $|\dot{\theta}_i|$, since we found that the phases converge faster than the locations in all relevant cases.
- *Splinted phase wave:* For the dynamic patterns, the speed of the swarmalators is not an indicator for convergence since the individual swarmalators are continuously moving around. Instead, it is the shape of the pattern that remains the same over time in a qualitative way. In particular, for the *splintered phase wave*, the swarmalators form an annulus that splits into a small number (about five to eight in our simulations) of clusters — the so-called splinters (see Fig. 1d). Within each cluster, the phases are almost homogeneous whereas they differ between clusters. We use this observation to form a criterion for convergence: Clusters are identified based on the spatial gap between them. If a certain threshold g^* is exceeded, we assume two separate clusters, whereas a single cluster is assumed otherwise. Then, we calculate the phase spread $s_j = \max_i \theta_i - \min_i \theta_i$ within each cluster. The system is considered to have converged if $\sum_j s_j < s^*$ with a small positive threshold value s^*.
- *Active phase wave:* The swarmalators in the *active phase wave* form an annulus similar to the *splintered phase wave* but without the gaps. Still, all swarmalators keep moving and continuously adjust their phases. We observe that the distribution of phases over the polar angle shows a characteristic picture: It forms a band with slope $+1$ or -1 and a width that approaches a typical value, which depends on N and other system parameters (see Fig. 1e). We consider this pattern to be converged once this band is formed and its width falls below a certain threshold b^*.

Based on these convergence criteria, we perform a series of numerical evaluations of the system (1) with random initial conditions using Euler's method with step size $\Delta t = 0.1$. For each p, we simulate a system of 100 swarmalators



(a) Average time until convergence (10 simulations per data point).



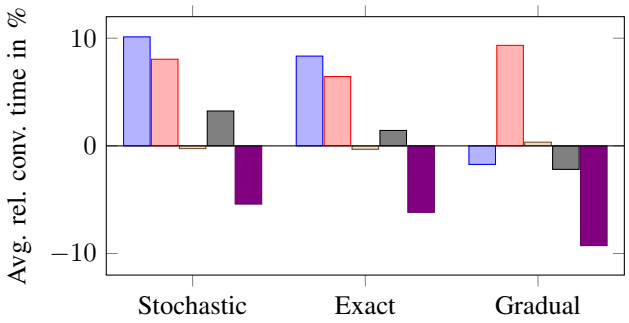
(b) Relative increase of convergence time in relation to $p = 1$ (30 simulations per data point).

Figure 2. Convergence time of the five patterns when varying the coupling probability p. Parameters are: 100 swarmalators, step size $\Delta t = 0.1$, memory initialized with 0.

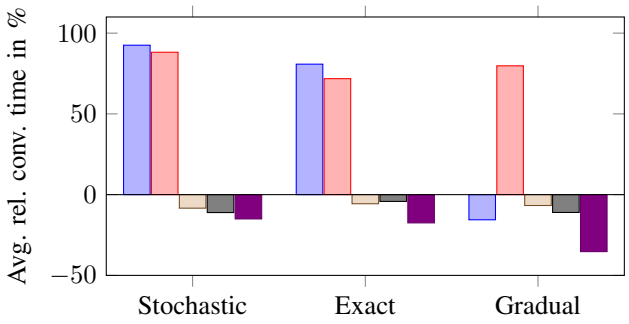
for all five patterns until it has converged. The resulting convergence time is averaged over 10 runs per data point.

From results shown in Fig. 2, we conclude that perfect coupling ($p = 1$) is not required for efficient convergence. On the contrary: for all five patterns, the value of p can be reduced significantly without suffering from relevant negative consequences, neither on final patterns nor on convergence time. This finding has the potential to greatly reduce the communication overhead. The observations are as follows:

- *Static patterns:* The coupling probability can be reduced down to $p \approx 0.05$ without any visible effect on the convergence time. This implies a reduction of the communication overhead by 95 %.
- *Splintered phase wave:* The convergence time of the *splintered phase wave* slightly increases for probabilities down to $p \approx 0.3$ and then increases significantly for smaller values. The sharp increase for low p-values illustrates that a dynamic pattern requires a higher update rate than a static pattern.
- *Active phase wave:* The convergence time of the *active phase wave* increases for decreasing p over the entire range. The delay is short for values down to $p \approx 0.4$ but increases significantly and more than in the other patterns for lower values.



(a) Results are averaged over $p = 0, \ldots, 1$.



(b) Results are averaged over $p = 0, \ldots, 0.1$.

Figure 3. Comparison between the convergence times of different memory initialization strategies for all five states (from left to right in each group of bars): *static sync*, *static async*, *static phase wave*, *splintered phase wave*, and *active phase wave*. Results are averaged over different coupling probabilities p. Initialization with 0 serves as baseline (100 %).

3.3. Initial Values of the Memory

In this work, we introduced a memory in each swarmalator to store the most recently received states of all other swarmalators (Section 3.1). An important question arises: How to initialize this memory? We suggest and compare four different approaches:

1) *All zero values.* The memories of all swarmalators are initialized with value zero. At the beginning, each swarmalator thus acts as if all others are located at the origin with a phase of 0. This approach applies a strong repelling force from the origin.
2) *Stochastic values.* The memories are initialized with random values, which are independently drawn from the distribution and support of the

actual initial locations and phases of the swarmalators. The reasoning for this approach is to avoid the repelling force of the all-zero approach by mimicking randomly distributed swarmalators. The probability distribution of the locations and phases must be known.

3) *True values.* Each memory is initialized with the actual location and phase of its swarmalator. From an implementation perspective, this approach resembles an initial exchange of all states prior to starting any movement or phase adaptation. The goal is to prevent collisions among swarmalators at an early stage of pattern formation.
4) *Gradual approach.* The memories are not initialized at the beginning, but only swarmalators that received state updates from other swarmalators store them in the memory. Initially, the sums in (1) are empty and the number of summands gradually increases with each received state update.

Fig. 3 shows the convergence times resulting from the different initialization strategies. Using the all-zero approach as baseline, we see the relative deviation from this baseline for all five patterns. Each bar shows an average over a certain range of p and 10 simulation runs per value of p.

All strategies lead to similar average convergence times if we average over the entire range of coupling probabilities $p = 0, \ldots, 1$ (Fig. 3a). An initialization with stochastic values leads to almost the same results as an initialization with the true values. In both cases, the memories are initialized with samples from the same probability distribution, and it seems that it hardly matters whether these samples correspond to the true values or not.

Clear differences arise if we focus on low coupling probabilities $p = 0, \ldots, 0.1$ (Fig. 3b). In such systems, the initial value of the memory is used for a longer time due to fewer interactions between swarmalators, leading to a stronger influence of the initialization on the convergence time. The *static async* pattern converges fastest with the all-zero initialization; the other initializations lead to average convergence times about twice as long. For the *static sync* pattern, the all-zero and the gradual approach work well. In all other patterns, all initialization strategies work about equally well. The *active phase wave* performs best when initialized with the gradual approach, leading to a reduction of the average convergence time of about $35\,\%$ compared to the all-zero approach.

Overall, there is no one-fits-all solution; determining the best initialization depends on the pattern. Still, the all-zero approach seems to be a good compromise considering its simplicity. In practice, aspects in addition to the convergence time need to be considered. For example, collision avoidance might favor an initialization with exact values.

3.4. Varying the Step Size

Fig. 4 shows the same convergence time simulation as Fig. 2a but now with the step size reduced to $\Delta t = 0.01$. A

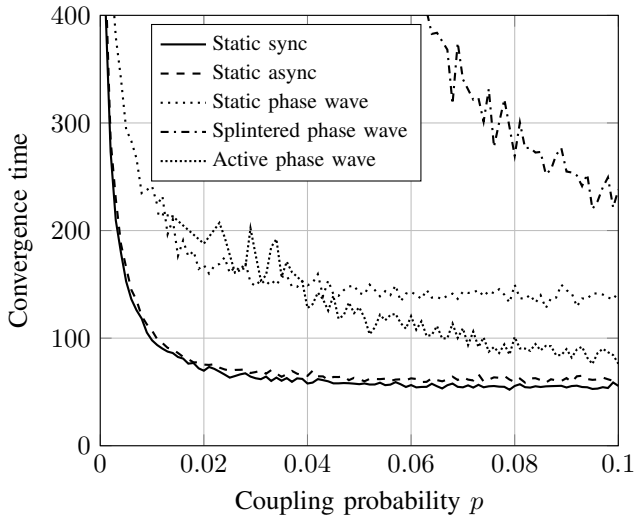


Figure 4. Average time until convergence (30 simulations per data point) for a reduced step size of 0.01. Memory is initialized with 0.

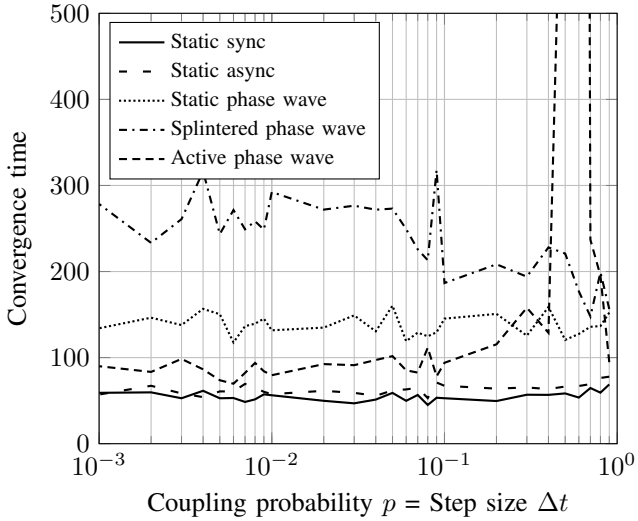
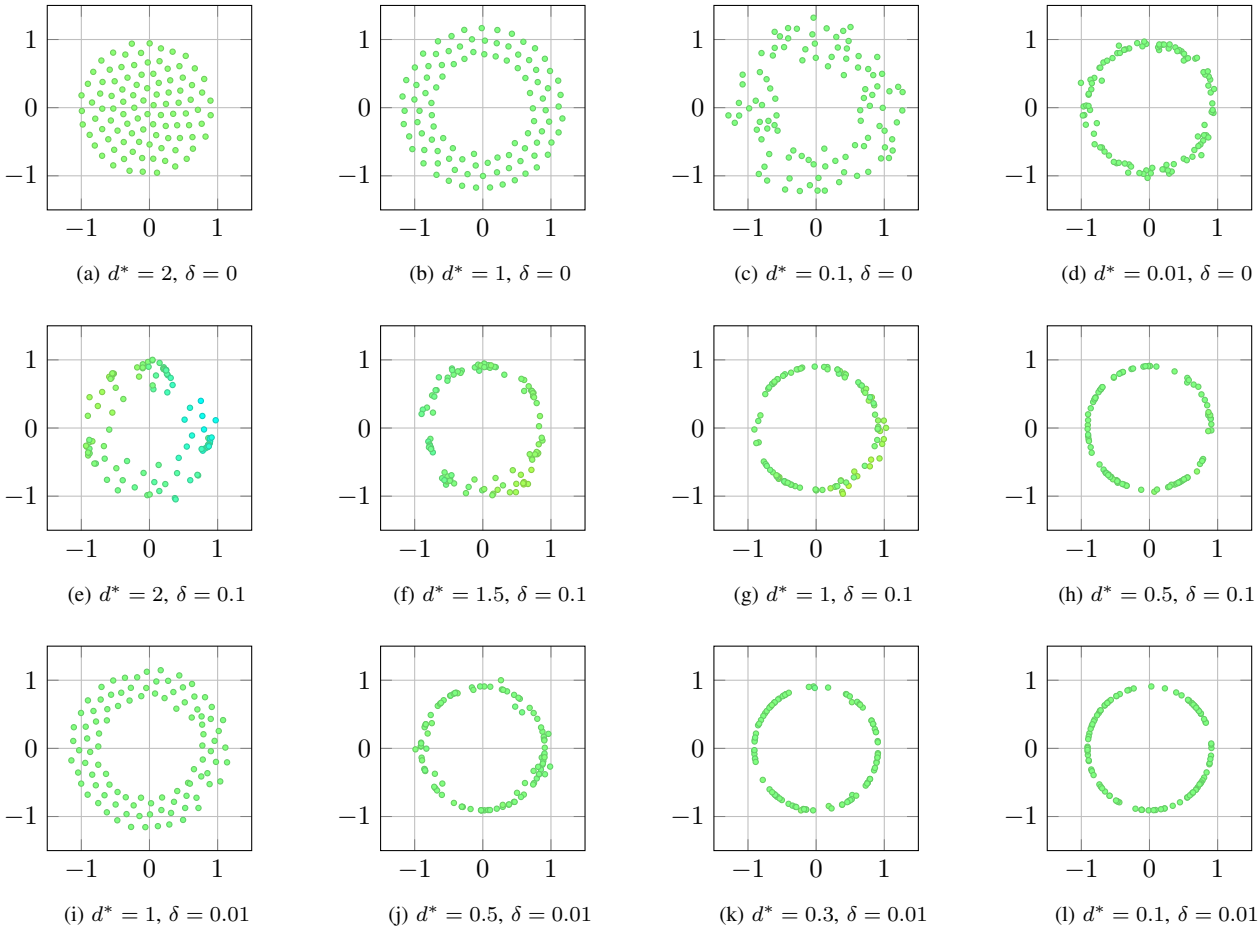


Figure 5. Comparison of the average time until convergence (10 simulations per data point) for different step sizes, while the communication rate in time is kept constant by keeping $p = \Delta t$. Memory is initialized with 0.

reduced step size yields more frequent updates of the states, as each swarmalator transmits with probability p in each iteration, which implies on average $p/\Delta t$ transmissions per time unit for each swarmalator. Since Δt is decreased by a factor of 10, we can also decrease p by a factor of 10, thus focusing on $p \leq 0.1$, to reach similar convergence times.

In Fig. 5, we adjust p and Δt simultaneously keeping their fraction constant (here at $p/\Delta t = 1$). For all patterns, the convergence time does not change beyond statistical fluctuation for small step sizes $\Delta t \leq 0.3$. Only for higher Δt, there is a sharp increase for the *active phase wave* at $\Delta t = 0.4$. This indicates that Δt is too high and the Euler method can no longer converge to the actual solution of the



(a) $d^* = 2$, $\delta = 0$ (b) $d^* = 1$, $\delta = 0$ (c) $d^* = 0.1$, $\delta = 0$ (d) $d^* = 0.01$, $\delta = 0$

(e) $d^* = 2$, $\delta = 0.1$ (f) $d^* = 1.5$, $\delta = 0.1$ (g) $d^* = 1$, $\delta = 0.1$ (h) $d^* = 0.5$, $\delta = 0.1$

(i) $d^* = 1$, $\delta = 0.01$ (j) $d^* = 0.5$, $\delta = 0.01$ (k) $d^* = 0.3$, $\delta = 0.01$ (l) $d^* = 0.1$, $\delta = 0.01$

Figure 6. *Static sync* patterns with 100 swarmalators with limited communication range. The initial communication range threshold d^* is reduced by δ once in each iteration, i.e., every $\Delta t = 0.1$ time units. Memory is initialized with 0. Patterns correspond to when convergence is reached.

system (1).

Decreasing Δt to very small values has no significant effect — neither on the convergence time nor on the resulting patterns (not shown). Note, however, that smaller step sizes imply a higher load on the CPU of the simulation hardware, as the number of iterations per time unit, i.e., the number of evaluations of (1), is $1/\Delta t$. A step size of $\Delta t = 0.1$ seems to be a good compromise between reliable convergence and reasonable computational effort in our setup.

3.5. Limited Communication Range

So far we analyzed swarmalators in the context of a fully meshed network (all-to-all coupling). The next step is to analyze a limited communication range, such that each swarmalator can only exchange states with its neighbors (similar to [19]). More specifically, we employ a time-varying range: each swarmalator starts with range d^* and then reduces this range in each iteration by δ. The idea behind this time-varying nature is to initially exchange many states on a global level and then reduce coverage to only receive updated states on a more local level.

Fig. 6 shows the results. One basic finding is: if the range is too small, a pattern different to the original pattern emerges. The first row shows four ranges $d^* = 2, 1, 0.1, 0.01$ that are time-constant ($\delta = 0$). For a range of 2, there is basically no constraint in communications since the diameter of the resulting pattern is about 2, thus the original pattern emerges. Reducing the range to 1, an annulus emerges instead of a disk, but the swarmalators are still positioned almost equidistantly on a grid-like pattern. Additional range reductions disturb the pattern further but eventually lead to a circle-like shape for a range of 0.01.

This change in pattern suggests that no state information of the far away swarmalators is ever received. Hence, we investigate how the pattern evolves over time if an initially high communication range is gradually reduced. Figs. 6e to 6h show the results for a change in range of $\delta = 0.1$ per iteration. Then, even for a starting range of 2, there is no emergence of the regular pattern: the circle does not fully form and even the phases cannot fully synchronize (see, e.g., the different colors in Fig. 6e), which corresponds to recent analytical results [19]. If we continue to reduce the range, the pattern further degrades and becomes almost a circle for very low ranges. Finally, for a more cautious range reduction

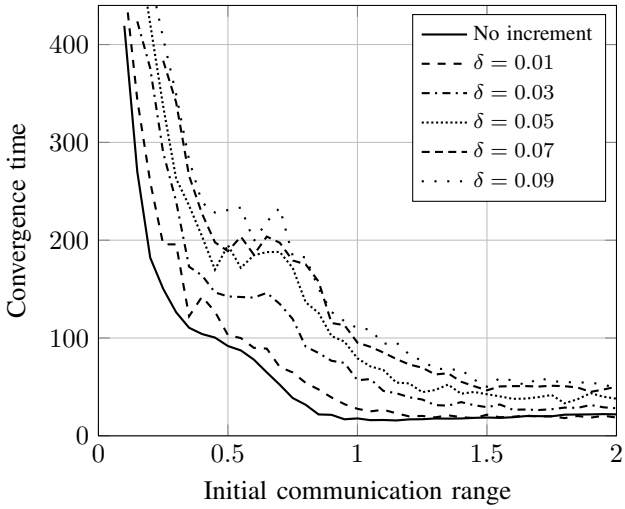


Figure 7. Average time until convergence (30 simulations per data point) for *static sync*. A maximum communication range is applied for which an initial range and a negative increment per iteration are varied. Convergence time is measured until some static pattern is reached. Parameters are: 100 swarmalators, initial area size is 2×2, step size 0.1. Memory is initialized with 0.

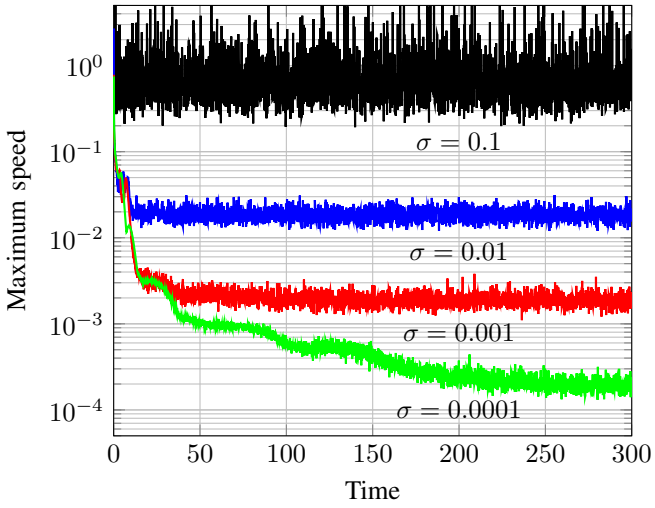


Figure 8. Maximum speed over all swarmalators over time for *static sync* for different inaccuracies σ for a single simulation run per σ-value.

with $\delta = 0.01$, shown in Figs. 6i to 6l, the pattern moves from an annulus ($d^* = 1$) to a perfect circle ($d^* = 0.1$), where all 100 swarmalators align on a single circle with small distances in between. This is only possible since the communication range quickly dies out and reaches zero after 10 iterations, so that all communication stops.

Next, we analyze the convergence time, which in this case is defined as the time it takes to reach one of the patterns shown in Fig. 6. Our results in Fig. 7 show that the convergence takes longer for lower communication ranges, in particular for ranges below 1. A faster decrease of the range also leads to a slower convergence. However, convergence occurs in almost all simulated setups within 10 000 iterations, indicating that a limited range does not prevent the system from converging. Although the resulting pattern is different, the convergence is always reached.

4. Further Issues in the Real World: Localization, Kinematics, and Size

4.1. Inaccurate Localization

Each swarmalator has to determine its location using some positioning method. The location information has a certain level of inaccuracy due to limitations of the method. We assume a system capable of absolute positioning, such as GPS, with an i.i.d. normally distributed positioning error with zero mean and variance σ^2 [20]. Although the inaccuracies may be small, they may possibly have unknown effects on the dynamics of the system. Hence, we investigate the convergence time for different magnitudes of random deviations overlaying the location information exchanged by the swarmalators. To do so, we replace x_j in the governing equations (1) by $\tilde{x}_j = x_j + (n_j, m_j)$ with the noise terms n_j and m_j being i.i.d. normally distributed with $N(0, \sigma^2)$ for $j = 1, \ldots, N$. Only the location is disturbed whereas the phase values are precise (as the phase does not have to be measured and is always precisely known).

A disturbed system needs new criteria for convergence. With the criteria used before, the system would never be considered as converged. In particular, for the static patterns, we defined convergence time as the time until the swarmalators' speed is below a certain threshold. However, this threshold might never be reached as the movement never stops due positioning errors on x_i. This effect is illustrated in Fig. 8, where the maximum speed over all swarmalators is plotted over time for different σ. The maximum speed decreases as the pattern emerges and finally reaches a floor level. This floor is at about 2σ in all shown cases except for $\sigma = 0.1$. In the latter case, no convergence can be observed, also judging from the evolving pattern (not shown). Hence, we redefine convergence to be the event when this floor is reached. This also enables a meaningful comparison of different σ-values. Fig. 8 shows that systems with smaller σ tend to take longer to reach their individual floor levels.

Fig. 9 shows the convergence time for *static sync* over the coupling probability p for different values of the positioning error standard deviation σ. The convergence threshold is $v^* = 1.5\,\sigma$. Several observations can be made: First, lower error variance leads to quicker convergence except for $\sigma = 0.1$. Second, in all simulated cases except $\sigma = 0.1$, the convergence time is almost independent of the coupling probability p for values $p > 0.05$. For lower values, the convergence time increases, which is in line with previous results without positioning errors (Fig. 2). Third, a high variance $\sigma = 0.1$ yields the fastest convergence of the whole plot for low p (due to a high threshold value that is quickly reached before actual convergence happens). Increasing p tends to slow down the process, especially above $p = 0.3$.

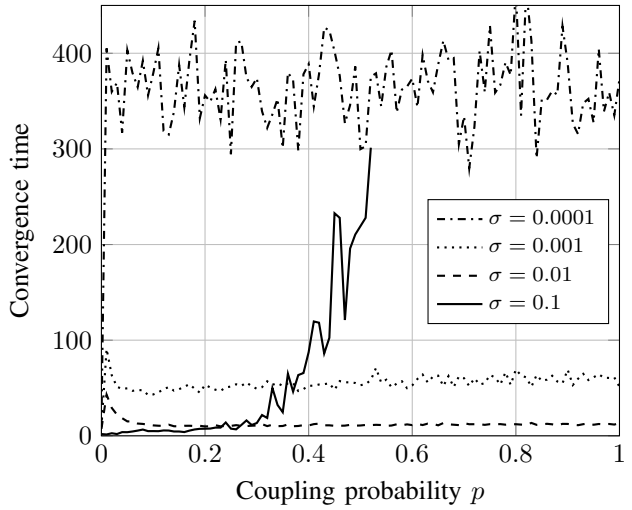


Figure 9. Average time until convergence for *static sync* (10 simulations per data point), when varying the positioning error variance. Parameters are: 100 swarmalators, step size 0.1. The convergence threshold is $v^* = 1.5\sigma$. Memory is initialized with 0.

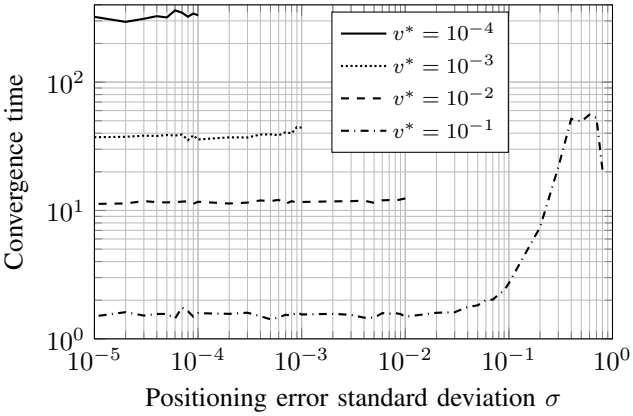


Figure 10. Average time until convergence for *static sync* (30 simulations per data point), when varying the positioning error standard deviation σ. The curves end at the point where the threshold is no longer reached. Parameters are: 100 swarmalators, step size 0.1. Memory is initialized with random values. The coupling probability is $p = 0.1$

For coupling probabilities above about 0.5, no convergence is observed within 10 000 steps (the curve ends there). This non-convergence can be explained as follows: Since every location update is subject to independent strong noise, the location information heavily fluctuates with every such update. In this situation, a rather constant (but inaccurate) location information leads to a quick convergence below the threshold (i.e., less movement). In contrast, regular updates of the location information (with different values) lead to quick fluctuations and much random movement, which in turn prevents the system from converging in the sense of our definition (also shown by the uppermost trace of Fig. 8).

Fig. 10 shows the convergence time over the error standard deviation σ for different convergence thresholds v^*. Two insights can be gained: First, the highest σ that can be tolerated without loosing convergence depends on the threshold: As long as $\sigma < v^*$, the system converges. Second, as long as the given threshold is ever reached, the convergence time is independent of σ. The only exception is when σ reaches a very high level and the threshold is also very high so it can be reached for such high σ: then the convergence times starts varying with σ. However, this is a very particular case.

4.2. Speed and Acceleration Limits

Mobile entities like robots and drones have a maximum possible speed and acceleration. The original model does not consider these limitations [16] but directly translates any force into a reciprocal movement. We propose the following extensions to incorporate the kinematic properties:

- *Limited speed.* Each node can change its position per time unit at most by a certain value $v_i = \frac{\|\dot{x}_i\|}{\Delta t} \leq v_\mathrm{m}$.
- *Limited acceleration.* Each node can change its speed at most by a certain limit $a = \dot{v}_i \leq a_\mathrm{m}$.

The three plots in Fig. 11 show the speed of the fastest swarmalator during the formation of the *static sync* pattern. Fig. 11a shows the highest speed for different speed limits v_m. At the beginning, the swarmalators utilize the maximum allowed speed v_m to approach their final locations, taking a variable amount of time. Then, they slow down and fine-adjust their locations before a steady state is reached. For different speed limits, once the speed is below its maximum, the slope of the slowdown is approximately the same for all cases, indicating that no impact beyond delaying the convergence is caused by the speed limit. This is confirmed by the resulting patterns (not shown).

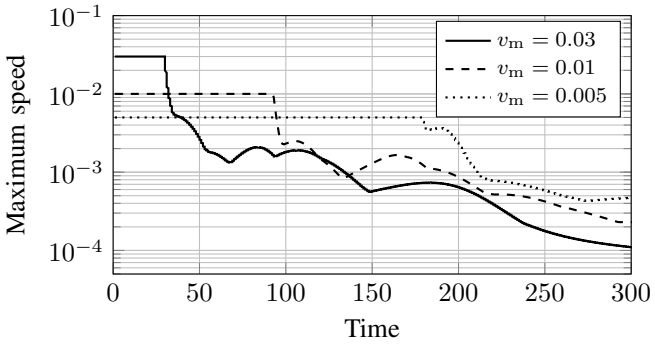
Fig. 11b shows the highest speed if an acceleration limit is applied. The swarmalators start with zero speed. To reach a meaningful speed, it takes an extended acceleration period. During this period, the slope of the curves is the maximum allowed acceleration. Note that the curves are bent due to the logarithmic scale — the speed increases linearly due to constant acceleration. Once the speed has reached its highest value, it immediately decreases (with the maximum allowed acceleration). At some point, the change in speed becomes so small that the acceleration limit loses its effect. Similar to the impact of the speed limit, the acceleration limit merely delays convergence but has no further impact on the emerging patterns.

Finally, Fig. 11c shows the combined effect of speed and acceleration limits. The speed limit cuts the curve of Fig. 11b and thus shifts the speed decay to the right. Again, no additional effects are to be seen.

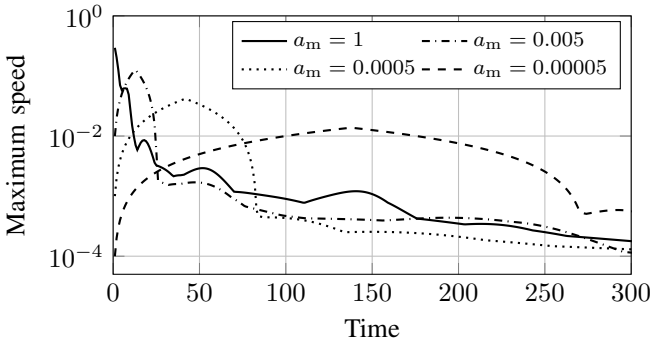
4.3. Physical Size and Scaling of Patterns

To avoid collisions between swarmalators [16], a minimum distance between any pair has to be imposed depending on the physical size. This can be achieved by modifying the repulsion term of the original model from

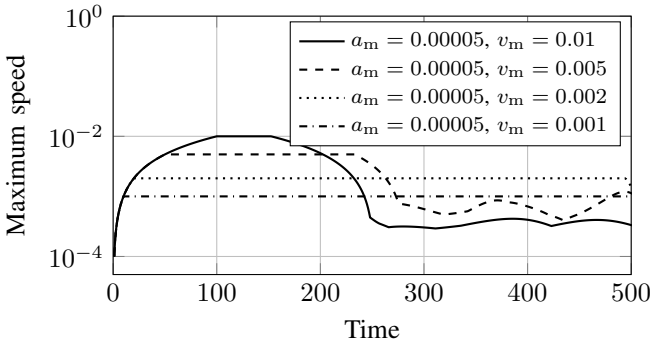
$$\frac{x_j - x_i}{\|x_j - x_i\|^2} \text{ to } \frac{x_j - x_i}{\min(\|x_j - x_i\|^2, (\|x_j - x_i\| - s)^2)}, \quad (2)$$



(a) Different speed limits v_m.



(b) Different acceleration limits a_m.



(c) Different speed limits v_m with an acceleration limit $a_\mathrm{m} = 0.00005$.

Figure 11. Development of the maximum value of the speeds of all swarmalators over time for *static sync* for a single simulation run per parameter combination.

where s denotes the physical size of the entity (approximated by a ball of diameter s). This modification does not completely avoid collisions as its effectiveness depends on the rate and accuracy of the location exchange.

Fig. 12 shows the smallest distance between any pair of swarmalators over time for different values of s. The first observation is that larger entities lead to higher pairwise distances, as the repelling forces at a given distance increase. Except for $s = 0.2$, the smallest distance is steady and above the value of s throughout the simulations. Interestingly, this is different for $s = 0.2$: First, the fluctuations of the smallest distance show that no convergence occurs, which can also be verified when observing the locations of the swarmalators throughout the simulation (not shown). Second, four spikes

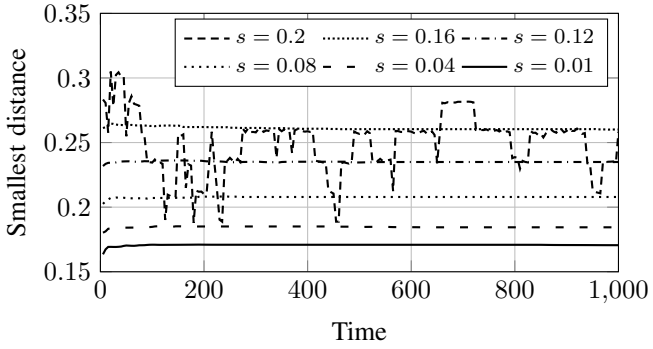


Figure 12. The smallest distance between swarmalators is plotted over time for *static sync*. The parameter s that introduces an additional repelling force with minimum distance s is varied.

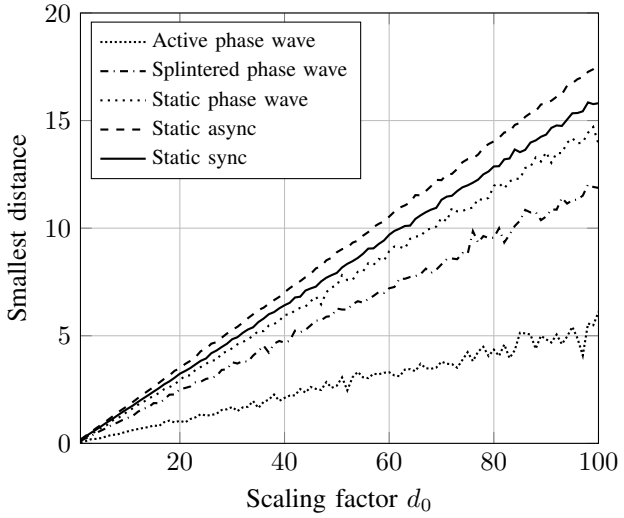


Figure 13. The smallest distance between pairs of swarmalators after 1000 iterations is plotted over time. Different metric scaling is applied.

can be seen in the plot where the smallest distance is below s. This is only possible due to the time-discrete evaluation of the equations by applying Euler's method. It is impossible that two entities are exactly at a distance s since the repelling force would approach infinity. However, it is possible that, within a single step, they jump over this singularity and reach a distance closer than s. This holds only for a short time, after which they would move away from each other due to the repelling forces. The explanation for this unexpected behavior is that the governing equations (1) determine the overall size of the resulting patterns depending on the parameters J and K. This size and the number of entities in turn determine a certain upper bound for the distances between neighboring entities to make them fit into the given space. If we superimposed a smallest distance higher than this upper bound, the system would be unable to support this smallest distance and hence it cannot converge. This problem can be overcome by re-scaling the overall pattern, as introduced in the following.

Fig. 13 applies a scaling factor to the distance metric

in (1) by substituting $\|\cdot\|$ by $\|\cdot\|/d_0^2$ for $d_0 = 1, \ldots, 100$. The results show that the smallest distance between any pair of entities in a converged state scales linearly with the distance metric, and so does the diameter of the resulting pattern (not shown). Interestingly, the smallest distance is quite different from pattern to pattern. For the *static async* almost a 3.5-times higher smallest distance is kept between pairs of entities compared to the *active phase wave*; this is related to the fact that enties in the *active phase wave* keep moving, resulting in an irregular pattern. The discrepancy between *static sync* and *static async* is not as easily explained; a main cause is the difference of the involved forces with equal versus different phases being next to each other. Note that these minimum distances are emerging without imposing a smallest distance on the swarmalators ($s = 0$ in Fig. 13).

5. Conclusions

In discretized swarmalator systems, the introduction of stochastic coupling in combination with storing the latest state information can significantly lower the communication overhead without severely slowing down the pattern formation process. A mere increase of the Euler step size beyond the value used in [11] would also achieve a reduction of the communication overhead but can lead to instabilities in the pattern formation. With stochastic coupling, all patterns still emerge with the communication rate reduced by $95\,\%$ without stability issues. The impact of the initial memory values and step size were studied for optimal system parametrization. Further insights were gained on real-world constraints: The process is robust against inaccurate localization as long as the errors are not too severe. Also a safety distance between swarmalators does not impact the convergence time. A speed or acceleration limit slows down the formation.

Acknowledgements

This work was partly funded by the Austrian Science Fund (FWF), grant *Self-organizing synchronization with stochastic coupling* (P30012).

References

[1] F. Dressler and O. B. Akan, "A survey on bio-inspired networking," *Computer Networks*, vol. 54, no. 6, pp. 881–900, 2010.

[2] Z. Shi, J. Tu, Q. Zhang, L. Liu, and J. Wei, "A survey of swarm robotics system," in *Advances in Swarm Intelligence* (Y. Tan, Y. Shi, and Z. Ji, eds.), (Berlin, Heidelberg), pp. 564–572, Springer, 2012.

[3] S. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar, "A survey on aerial swarm robotics," *IEEE Trans. on Robotics*, vol. 34, no. 4, pp. 837–855, 2018.

[4] M. Senanayake, I. Senthooran, J. C. Barca, H. Chung, J. Kamruzzaman, and M. Murshed, "Search and tracking algorithms for swarms of robots: A survey," *Robotics and Autonomous Systems*, vol. 75, pp. 422–434, 2016.

[5] A. Tahir, J. Bling, M.-H. Haghbayan, H. T. Toivonen, and J. Plosila, "Swarms of unmanned aerial vehicles – a survey," *Journal of Industrial Information Integration*, vol. 16, p. 100106, 2019.

[6] C. Virágh, G. Vásárhelyi, N. Tarcai, T. Szrényi, G. Somorjai, T. Nepusz, and T. Vicsek, "Flocking algorithm for autonomous flying robots," *Bioinspiration & Biomimetics*, vol. 9, p. 025012, May 2014.

[7] S. H. Strogatz, *Sync: How Order Emerges from Chaos in the Universe, Nature, and Daily Life*. Hachette Books; Reprint Edition, 2004.

[8] Y. Kuramoto, "Collective synchronization of pulse-coupled oscillators and excitable units," *Physica D: Nonlinear Phenomena*, vol. 50, no. 1, pp. 15–30, 1991.

[9] R. E. Mirollo and S. H. Strogatz, "Synchronization of pulse-coupled biological oscillators," *SIAM Journal on Applied Mathematics*, vol. 50, no. 6, pp. 1645–1662, 1990.

[10] R. Pagliari and A. Scaglione, "Scalable network synchronization with pulse-coupled oscillators," *IEEE Trans. on Mobile Computing*, vol. 10, no. 3, pp. 392–405, 2011.

[11] K. P. O'Keeffe, H. Hong, and S. H. Strogatz, "Oscillators that sync and swarm," *Nature Communications*, vol. 8, no. 1, p. 1504, 2017.

[12] K. P. O'Keeffe, J. H. Evers, and T. Kolokolnikov, "Ring states in swarmalator systems," *Physical Review E*, vol. 98, no. 2, p. 022203, 2018.

[13] A. Barciś and C. Bettstetter, "Sandsbots: Robots that sync and swarm," *IEEE Access*, vol. 8, pp. 218752–218764, Dec. 2020.

[14] J. Klinglmayr, C. Kirst, C. Bettstetter, and M. Timme, "Guaranteeing global synchronization in networks with stochastic interactions," *New Journal of Physics*, vol. 14, July 2012.

[15] K. O'Keeffe and C. Bettstetter, "A review of swarmalators and their potential in bio-inspired computing," in *Proc. SPIE Micro- and Nanotechnology Sensors, Systems, and Applications XI*, (Baltimore, MD, USA), p. 109822E, Apr. 2019. Invited paper.

[16] A. Barciś, M. Barciś, and C. Bettstetter, "Robots that sync and swarm: A proof of concept in ROS 2," in *Proc. Intern. Symp. on Multi-Robot and Multi-Agent Systems (MRS)*, (New Brunswick, NJ, USA), pp. 98–104, Aug. 2019.

[17] B. N. Biswas, S. Chatterjee, S. P. Mukherjee, and S. Pal, "A discussion on Euler method: A review," *Electronic J. of Mathematical Analysis and Applications*, vol. 1, no. 2, pp. 294–317, 2013.

[18] L. Bhatia, I. Tomić, A. Fu, M. Breza, and J. A. Mccann, "Control communication co-design for wide area cyber-physical systems," *ACM Trans. Cyber-Phys. Syst.*, vol. 5, Jan. 2021.

[19] H. K. Lee, K. Yeo, and H. Hong, "Collective steady-state patterns of swarmalators with finite-cutoff interaction distance," *Chaos*, vol. 31, no. 3, p. 033134, 2021.

[20] C. C. J. M. Tiberius and K. Borre, "Are GPS data normally distributed?," in *Geodesy Beyond 2000* (K.-P. Schwarz, ed.), (Berlin, Heidelberg), pp. 243–248, Springer, 2000.