Evolving Aggregation Behaviors in Multi-Robot Systems

Supplementary Material

Melvin Gauci, Jianing Chen, Tony J. Dodd and Roderich Groß

1 Grid Search Visualization and Analysis

The resolution used in the grid search was twenty-one settings per parameter:

$$\{-1.0, -0.9, \ldots, 0.0, 0.1, \ldots, 0.9, 1.0\}, \tag{1}$$

where -1.0 and 1.0 correspond to the maximum backward and forward rotation speeds of the wheel, respectively. Therefore, $21^4 = 194481$ controllers were tested in total. Each controller was evaluated 100 times using Eq. (2) in [1] with different seeds (i.e. different initial configurations of robots), with the set of seeds being identical for each controller. The fitness of each controller was recorded as the mean fitness of the 100 evaluations:

$$\bar{F}(\mathbf{x}) = \frac{1}{100} \sum_{k=1}^{100} (\mathbf{x}, \psi_k) \tag{2}$$

The fitness landscape with the reactive controller is 5-dimensional (4 parameters plus fitness), and therefore cannot be visualized as-is. Therefore, in order to obtain a reasonable visualization, each possible combination of two parameters (${}^4C_2 = 6$ combinations in total) was considered separately, with the fitness at each point in the subspace being taken as the maximum possible fitness achievable with the remaining two parameters as degrees of freedom (within the resolution of the grid search). For example, for the sub-space $\left(s_l^0, s_r^0\right)$, the fitness $\bar{F}^*\left(s_l^0, s_r^0\right)$ at each point was calculated as:

$$\bar{F}^*\left(s_l^0, s_r^0\right) = \max_{s_l^1, s_r^1} \bar{F}\left(s_l^0, s_r^0, s_l^1, s_r^1\right),$$

and similarly for the remaining two-parameter sub-spaces. Fig. 1 shows these six landscapes as color maps. As the robots used here are circular and symmetrical, if the speed signals from the controller to the left and the right wheels of *every* robot had to be inverted, the resulting behavior will be unaffected, other than that it would also be 'inverted' in space. This is reflected in the facts that (i) the fitness landscapes on

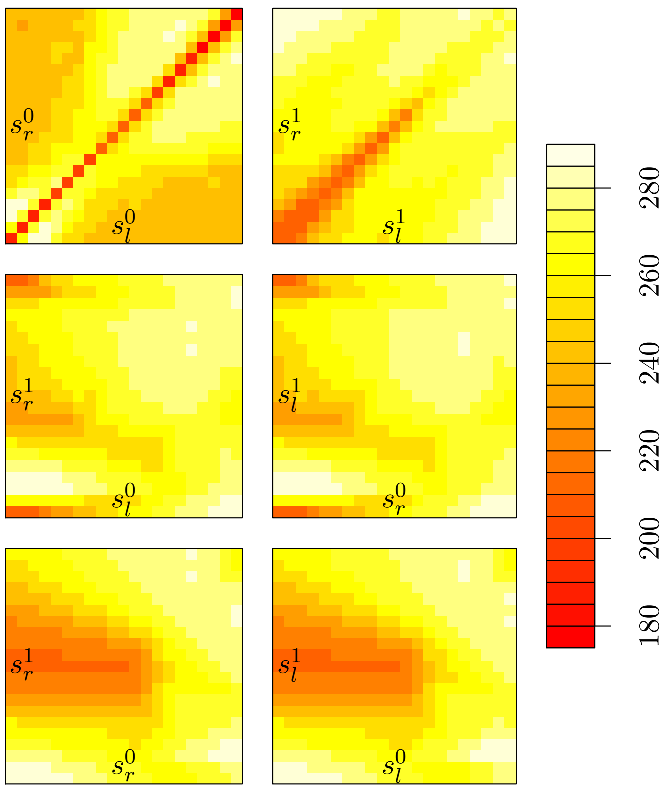

Figure 1: A visualization of the fitness space with the reactive controller, as explored by the grid search. See the text for details.

$\left(s_l^0, s_r^0\right)$ and $\left(s_l^1, s_r^1\right)$ are nearly symmetrical along the diagonals $s_l^0 = s_r^0$ and $s_l^1 = s_r^1$, respectively, and (ii) the fitness landscapes on the sub-spaces $\left(s_l^0, s_r^1\right)$ and $\left(s_r^0, s_l^1\right)$ are nearly identical to each other, as are those on the sub-spaces $\left(s_r^0, s_r^1\right)$ and $\left(s_l^0, s_l^1\right)$.

The fitness landscape on the sub-space $\left(s_l^0, s_r^0\right)$ (top left plot in Fig. 1) shows a marked 'valley' along the diagonal defined by $s_l^0 = s_r^0$, implying that having the robots move in a straight line when they do not perceive another robot leads to a poor aggregation performance. This can be interpreted intuitively, because robots that happen to be on the periphery of the random initial configuration, and facing 'outwards', will diverge from the rest of the group, without a possibility of ever changing their course (assuming an unbounded environment). The fitness landscape on the sub-space $\left(s_l^1, s_r^1\right)$ also shows a 'valley' along the diagonal defined by $s_l^1 = s_r^1$, implying that it is not optimal for the robots to hone in a straight line onto a perceived robot. This could be considered somewhat counter-intuitive, because a hand-designed controller would probably opt for this option, at least as an initial guess.

From Fig. 1, it is clear that the fitness landscape is not uni-modal, but rather consists of a number of peaks and troughs. This justifies the use of an evolutionary algorithm to synthesize (i) a reactive controller with a higher resolution, and (ii) a recurrent controller,

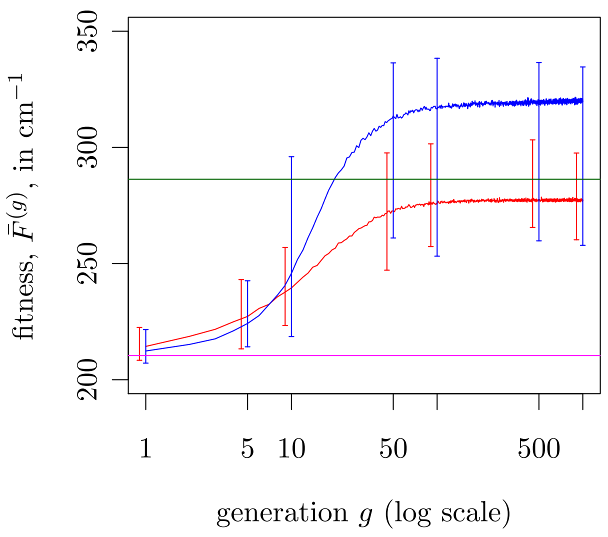

Figure 2: This plot shows the evolutionary dynamics for two sets of 100 evolutionary runs: one with a recurrent controller (blue) and one with a reactive controller (red). The horizontal axis shows the generation, g, (with a log scale), while the vertical axis shows the mean population fitness, $\bar{F}^{(g)}$, as defined in Eq. (3). The curves show the mean $\bar{F}^{(g)}$ over the 100 runs, while the vertical bars show the minimum and maximum value of $\bar{F}^{(g)}$ in generations $\{1, 5, 10, 50, 100, 500, 1000\}$. The horizontal green line shows the best value of $\bar{F}^{(g)}$ found by the grid search with a reactive controller. The horizontal magenta line shows the expected value of $\bar{F}^{(g)}$ for robots that do not move throughout the simulation (i.e. retain their initial configuration).

which is impractical to synthesize using a grid search as it consists of 8 unbounded, real-valued parameters.

2 Evolutionary Dynamics

In order to monitor the dynamics of the evolutions, the $\mu = 15$ selected individuals in every generation of every evolution were evaluated one additional time each with different initial configurations of robots. The fitness of the population at generation g was computed as the average of these $\mu = 15$ evaluations:

$$\bar{F}^{(g)} = \frac{1}{\mu}\sum_{k=1}^{\mu} F\left(\mathbf{x}_k^{(g)}, \psi_k\right) \tag{3}$$

controller	s_l^0	s_r^0	s_l^1	s_r^1
grid search reactive	-0.80	-1.00	1.00	-1.00
evolved reactive	-0.74	-1.00	0.99	-1.00
evolved recurrent	$\rightarrow -1.00$	$\rightarrow -0.85$	$\rightarrow -1.00$	$\rightarrow 1.00$

Table 1: The top two rows of this table show the parameters of the best-performing reactive controllers synthesized with the grid search (top row) and the evolutionary algorithm (middle row). The bottom row shows the values of the wheel speeds to which the outputs recurrent controller converge if a constant input ($I = 0$ or $I = 1$)is applied successively for a number of control cycles.

Fig. 2 shows the dynamics of $\bar{F}^{(g)}$ for the two sets of evolutions (see caption for details). It is clear that the dynamics reach a near-steady-state by the 1000th generation. The mean value of $\bar{F}^{(g)}$ with a recurrent controller exceeds the one with a reactive controller after around 10 generations. In the final generation, the worst-performing recurrent controller has a lower fitness than the worst- performing reactive controller; this is because, as explained in [1], one of the evolutions with a recurrent controller led to a circle-forming controller (see Fig. 2(b) in [1]).

3 Controller Analysis

The top two rows in Table 1 show the parameters of the best- performing reactive controllers synthesized by the grid search (top row) and the evolutionary algorithm (middle row). It is evident that the evolutionary algorithm located essentially the same controller; indeed, the only parameter that is significantly different between the two controllers is the speed of the left wheel when $I = 0$, s_l^0, which has a value of -0.80 for the controller synthesized by the grid search, and a value of -0.74 for the evolved controller. Interestingly, this small difference in the parameters accounts for a significant difference in the aggregation performance of the two controllers (see Fig. 2(b) in [1]). The reactive controllers operate as follows. When a robot does not perceive another robot (i.e. when $I = 0$), it moves *backwards* with a curved trajectory. When a robot, on the other hand, perceives another robot, it stops moving (as $s_l^1 \approx s_r^1$) and turns on the spot in a clockwise direction.

The bottom row of Table 1 corresponds to the best-performing recurrent controller synthesized by the evolutionary algorithm. The values in the table show the values of the wheel speeds that the controller outputs converge to if the same input is applied for a consecutive number of control cycles (hence the arrows in the table). For instance, if a constant input $I = 0$ is presented to the network, then, after a number of control cycles, the speed of the left wheel of the robot converges to -1.00, and so on. When one looks at the recurrent controller in this manner, it is clear that its basic dynamics are similar to those of the reactive controllers, but with the left and the right speeds interchanged (recall that since the robots are symmetric in shape, such an interchange

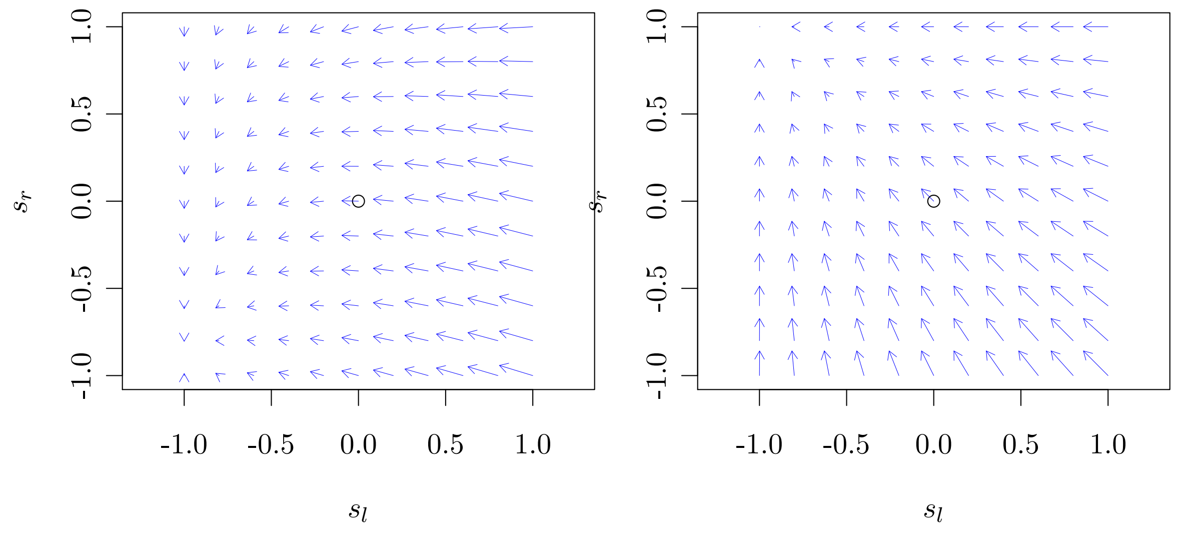

Figure 3: These plots show the internal dynamics of the best-performing recurrent controller synthesized with the evolutionary algorithm. The left plot is for the input $I = 0$, while the right plot is for the input $I = 1$. Each arrow shows in which direction the state of the network, (γ_1, γ_2) moves, and the length of each arrow is proportional to the distance that the state moves in that direction. Such a plot is sometimes called a "quiver plot".

leads to an essentially identical behavior). This shows that the memory inside the recurrent controller is responsible for its superior performance (as shown in Fig. 2(b) in [1]. Therefore, one can conclude that, while memory is not an essential component for aggregation of robots with binary sensors, it can improve the performance of a controller in terms of the speed of aggregation. Figure 3 shows the internal dynamics of the recurrent controller (see the caption for details).

4 Fitness Dynamics

Figure 4 shows the dynamics of aggregation with 100 robots. 100 simulations were run with different seeds (i.e. different initial configurations of robots), and the value of $1/d^{(t)}$ was recorded at each time step. Additionally, the time it took for the robots to first form a single cluster (according to the definition in Footnote 2 in [1]) was recorded. The dynamics of all runs reach a near steady-state within 1800 seconds (30 minutes).

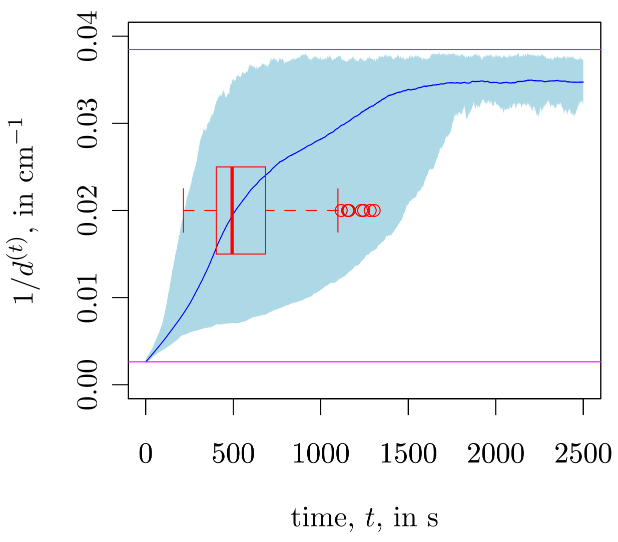

Figure 4: This plot shows the dynamics of aggregation with 100 robots. The horizontal axis represents the time, t, while the vertical axis shows a measure of aggregation, $1/d^{(t)}$. The blue curve shows the mean aggregation measure over 100 runs with different initial configurations, while the shaded light blue area shows the range (minimum, maximum) of the aggregation measure across the 100 runs. The horizontal red box plot shows the times taken by the 100 runs to achieve a single cluster of robots. The horizontal magenta lines show the random (lower) and the maximum (upper) values of $1/d^{(t)}$ with 100 robots.

5 The Effect of Sensing Noise

False negative and false positive noise on the binary sensor are described in Section 4 of [1]. Dual noise is a combination of these two noises in equal proportions. Figure 5 shows the effect of these three types of noise on aggregation performance (see the caption for details). False positive noise is detrimental to the performance. False negative noise increases the performance up to a certain level (this corresponds to the 'bowl' shape in Fig. 3(b) in [1]). Dual noise is also detrimental to aggregation performance, but less so than false positive noise alone.

6 Scalability Study

Figure 6 shows the results of a scalability study with up to $N = 1000$ robots (see the caption for details). It is worth noting that 1000 robots consistently aggregated into a

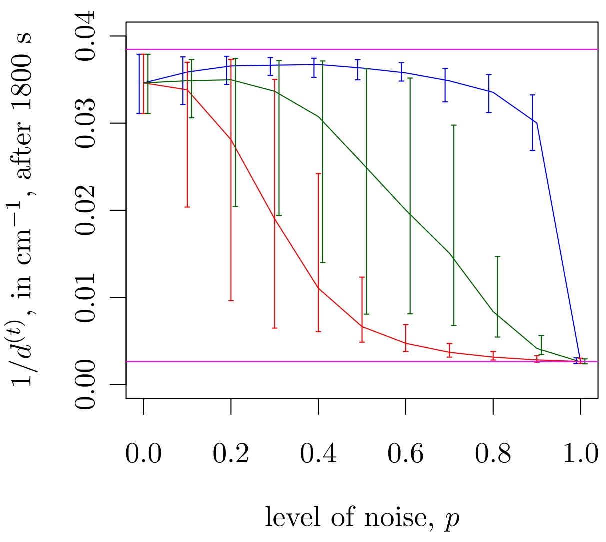

Figure 5: This plot shows the effect of sensing noise on aggregation performance. The blue, red and green curves correspond to false negative, false positive and dual noise, respectively. The horizontal axis represents the time, t, while the vertical axis shows a measure of aggregation, $1/d^{(t)}$. The horizontal magenta lines show the random (lower) and the maximum (upper) values of $1/d^{(t)}$ with 100 robots. The curves show the mean value of $1/d^{(t)}$ across 100 runs with different initial configurations of robots, while the vertical show the minimum and the maximum value of $1/d^{(t)}$ across the 100 runs.

single cluster across 100 runs with randomized initial configurations.

References

[1] M. Gauci, J. Chen, T. J. Dodd and R. Groß, *Evolving aggregation behaviors in multi-robot systems*, submitted to DARS 2012.

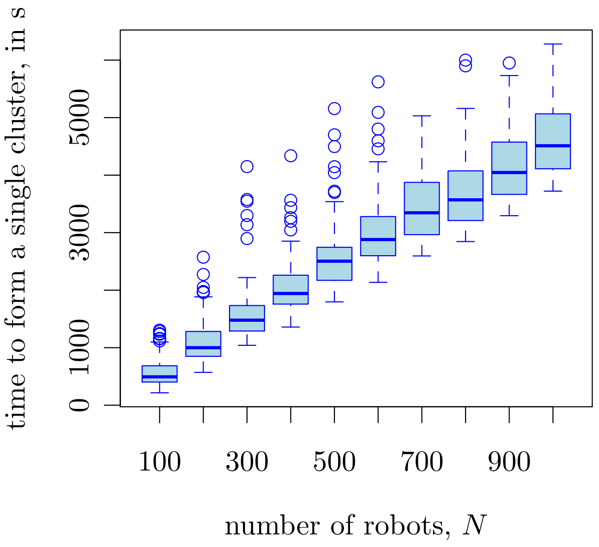

Figure 6: This box plot shows the time that it takes $N = \{100, 200, \ldots, 1000\}$ robots to aggregate into a single cluster. For each value of N, 100 simulations were run with different initial configurations of robots.