Self-organized Construction by Population Coding

Michael Nieß
Institute of Computer Engineering,
University of Lübeck
Lübeck, Germany

Heiko Hamann
Institute of Computer Engineering,
University of Lübeck
Lübeck, Germany
hamann@iti.uni-luebeck.de

***Abstract*—The automatic generation of robot controllers by machine learning or evolutionary computation is still challenging and even more so for collective robotics. We follow the recently proposed paradigm of 'population coding' to compose robot swarms for collective construction. We define a controller template as finite state machine, enumerate a finite number of specified robot controller types to choose from, and use evolutionary robotics to evolve effective homogeneous and heterogeneous compositions of robot swarms using selections of these controllers. Besides an objective for solving the actual construction task we also add objectives for subtasks, and to minimize the number of different chosen robot types. For three variants of a collective construction task we find effective solutions with both homogeneous and heterogeneous swarms.**

***Index Terms*—swarm robotics, population coding, heterogeneous swarm, self-assembly**

I. Introduction

The automatic generation of robot controllers requires methods of machine learning or evolutionary robotics [1]. Generating controllers for multi-agent or collective systems is even more challenging. Multi-agent learning suffers from the combinatorial explosion of possible agent-action pairs [2] and similarly for the application of evolutionary computation to swarm robotics [3], that is, evolutionary swarm robotics [4].

Here, we follow an alternative approach: the population coding paradigm [5]. The idea is to define a finite set of robot controller types from which we then compose the robot swarm that solves the desired task. The robot controller types can be seen as robot types with hardwired logics. The only remaining freedom for programming is on the swarm level. In analogy to baking a cake, the desired swarm behavior is generated by choosing the right quantities of the right ingredients. For a swarm of size N, we can form homogeneous swarms by choosing N-times the same robot type or heterogenenous swarms by choosing up to $N-1$ of one type and at least one different robot type. Correctly choosing numbers and types is seen as an optimization problem, that we solve by evolutionary computation. Hence, population coding can also be a method to usefully limit the search space in an evolutionary swarm robotics approach. Related studies on heterogeneous swarms are: evolution of task specialization [6], [7] and formalization of heterogeneity [8]. The main contribution here is the novel application of population coding to collective construction.

II. Tasks and environment

A swarm of $N = 10$ simulated robots operates in a $10\,\mathrm{m} \times 10\,\mathrm{m}$ rectangular arena surrounded by walls (see Fig. 1).

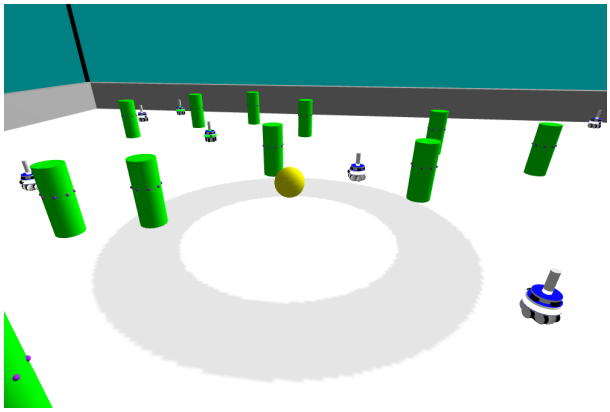


Fig. 1: Screenshot of the simulation: robots (blue) pull cylinders (green) into the target zone (gray) to encircle the light source (yellow) in the middle.

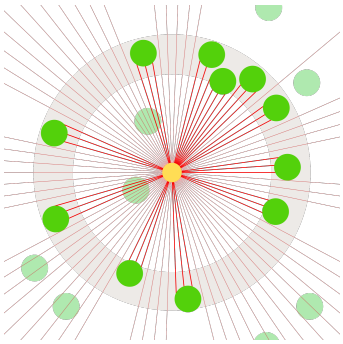


Fig. 2: Task performance is measured by coverage. 360 raycasts and only cylinders within the gray ring are considered. The percentage of rays hitting cylinders is coverage.

20 cylinders with a radius of $0.1\,\mathrm{m}$ serve as building blocks. There is a light beacon in the middle of the arena. The robots have a light sensor and use the light beacon for navigation. The desired shape of the structure that needs to be built is defined by a ring-shaped target zone marked in gray and encircling the light beacon (ring at a distance from the beacon between 0.75 m and 1.25 m).

The task for the robots is to build a shelter around the light beacon by pulling cylinders on the target zone. Task performance is quantified by *coverage*, that is, how much light is measured outside of the target zone (see Fig. 2). If there is no

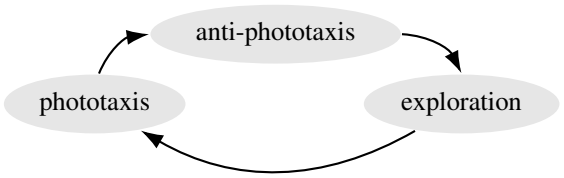


Fig. 3: Finite state machine of the robot controller template.

input	parameter	range	resolution	# bits
duration T	θ_T	$[0\,\mathrm{s}, 10\,\mathrm{s}]$	5	3
light L	θ_L^{low}	$[0, 1]$	5	3
light L	θ_L^{high}	$[0, 1]$	5	3
cylinder detection D	θ_D	$\{1; 0; -1\}$	3	2
-	θ_A	$\{0; 1\}$	2	1

TABLE I: Parameters used to determine the transition of the finite state machine controller template.

light measured outside of the target zone, then the robots have built a perfect ring structure (100% coverage). The cylinders are taller than the robots and the light beacon is positioned higher than the robot height. Coverage is measured by ray tracing where cylinders outside of the gray ring are ignored (360 rays, gap between rays at ring about 0.018 m).

We study three variants of a construction task: basic scenario, stay-in scenario, and stay-out scenario. In the basic scenario task performance exclusively relies on coverage. If at the end of the evaluation there are many cylinders on the gray ring and they block a lot of light, then the swarm receives a high reward.

For the stay-in scenario we introduce a second objective: stay close to the light source. We use the distance d between light source and robot to calculate a reward $r = \frac{1}{d^2}$ for each robot in each simulation step. We use the sum over all time steps and all robots as additional objective. The rationale is to increase the task complexity because searching for unused cylinders and staying at home are contradicting subtasks. A solution could be a heterogeneous swarm with one subpopulation specializing in searching/pulling cylinders and another one staying close to the light source.

For the stay-out scenario we invert the reward. The objective is to stay as far away from the light source as possible. The rationale is again to increase the task complexity and pulling cylinders towards the light contradicts the stay-out objective.

III. Approach

A. Robot controller template

Following the population coding paradigm [5] we define first a simple finite state machine as a template robot controller that roughly covers the required behavioral features. We define and implement three states covering primitive behaviors: *exploration*, *phototaxis*, and *anti-phototaxis* (see Fig. 3). In addition, we define two actions that a robot can trigger by state transitions: *pickup* and *drop* of cylinders.

In state *exploration* a robot does a random walk and avoids obstacles. In moments when no obstacle needs to be avoided, the robot moves in straight lines and chooses a new direction every two seconds with a random angle in the range of $[-45°, 45°]$ from its current direction. This enables the robots to move smoothly in the arena without abrupt turns. In state *phototaxis* a robot moves towards the light source and avoids obstacles. The given scenario may result in situations where the light is covered by cylinders, leaving the robot without orientation. In these cases the robot switches to an exploration behavior until it senses light and is able to move towards it again. In state *anti-phototaxis* the robot behaves similarly except for moving away from the light.

When the *pickup* action is triggered by a transition, it is only executed if the robot senses a cylinder nearby and is currently not dragging another cylinder. In these cases, the robot moves towards the cylinder and grabs it. This finishes the *pickup* action and the next state is entered. The robot now pulls the cylinder and moves slower until it drops the cylinder again. Similarly, the *drop* action is only triggered if the robot pulls a cylinder. If executed, the gripper releases the cylinder. During the time a robot pulls a cylinder its avoidance behavior is turned off. Otherwise robots would not be able to place cylinders close to other cylinders.

Each of the three transitions of the finite state machine (Fig. 3) has the same structure. It processes three sensor inputs and takes five configuration parameters. The transitions between the three states rely on the light intensity L, duration in current state T, and cylinder detection D. We combine them with five parameters as seen in Tab. I: minimal time θ_T staying in the current state, low light threshold θ_L^{low}, high light threshold θ_L^{high}, cylinder detection θ_D, and instant action θ_A. If the light parameters are mutually exclusive ($\theta_L^{\text{high}} < \theta_L^{\text{low}}$), we do not consider them.

We distinguish three states concerning cylinder detection θ_D. There has to be a cylinder detected (1), there must not be a cylinder detected (-1), and an indifferent state (0), that always evaluates true regardless of any cylinder detection.

The instant action parameter θ_A is only binary and indicates whether a cylinder should be picked up (1) or dropped (0). We do not define a parameter for 'no instant action,' as it is not required. The same instant action may be executed repeatedly without any effect.

As the parameters are encoded digitally, possible values for parameters are defined by a range and resolution. Values are equally spread over the whole range. Parameters θ_T, θ_L^{high}, and θ_L^{low} have a range of $[0, 1]$ and a resolution of five giving values $\{0; 0.25; 0.5; 0.75; 1\}$.

Combining all of the above, we get a logical expression δ determining whether a transition is triggered

$$\begin{aligned}\delta(T, L, D) =& (T > \theta_T) \\ & \wedge ((\theta_L^{\text{low}} < L < \theta_L^{\text{high}}) \vee (\theta_L^{\text{high}} < \theta_L^{\text{low}})) \\ & \wedge (\theta_D = 0 \vee D = \theta_D) \end{aligned} \quad (1)$$

Based on each parameter's resolution, we calculate the amount of possible transition configurations as $5 \times 5 \times 5 \times$

$3 \times 2 = 750$. Given the three transitions in our finite state machine, there are 750^3 different robot controller types. For a heterogeneous swarm we have 750^3 options to choose from. For a swarm size of $N = 10$, we get a set with an upper bound of $(750^3)^{10}$ possible heterogeneous swarms.

B. Homogeneous versus heterogeneous swarms

Given the general finite state machine with its five parameters per transition (Tab. I), we have to decide whether we want to implement a homogeneous swarm with one robot type using a common controller or a heterogeneous swarm with several different robot controllers. For our swarm of N robots, we have to select controllers, the number M of robot controller types that we want to use, and how many of each of them. Swarm robotics is usually defined based on homogeneous swarms, however, recent research has shown that heterogeneous swarms may have advantages, too [9], [10]. We define a swarm configuration $C = (N, M, c_1, c_2, \ldots, c_M, n_1, n_2, \ldots, n_M)$, for used controllers c_i of type i and number n_i of robots that use controller i with $\sum_i n_i = N$. In general the resulting swarm behavior of a swarm configuration C is difficult to predict even for homogeneous swarms ($M = 1$). With a mathematical micro-macro link [11], however, we could predict the macroscopic effect of the microscopic robot behaviors, define an analytic performance measure $P(C)$ and formulate the problem of finding the best configuration as a regular optimization problem. As we have no micro-macro link and no easily defined performance measure, here we need to evaluate configurations in simulation, similarly to the leafcutter ants example by Hamann et al. [5]. We use the swarm simulator ARGoS [12] to evaluate swarm configurations.

C. Multi-objective optimization, sparsity, and hypervolume

The number of $(750^3)^{10}$ possible heterogeneous swarms cannot be tested in a brute force attempt. Each evaluation is based on a costly multi-robot simulation run. We need to apply a more sophisticated optimization technique. Here, we use a simple genetic algorithm [13]. As fitness function we could simply assess the efficiency of the finished construction according to its light coverage (and cumulative distance to the light source for the stay-in and stay-out scenarios). However, there is another option for an objective: sparsity. With sparsity we mean the heterogeneity of the swarm, that is, the number M of used different controllers and how different the used controllers actually are. Minimizing M is useful because each additional controller comes with overhead. More controllers also reduce the redundancy in the swarm and hence limit the swarm's robustness. We try to use as few different robot types as possible.

There are different ways to define sparsity [14]. We could use the number of used different robot types which corresponds to the ℓ_0-norm. However, we prefer a more precise measure of sparsity. We sum Hamming distances between all robot genes (cf. [8]). Assuming robots with similar genes behave similarly, a broken robot could be replaced by a similar robot. Hence, we would have a higher degree of robustness. We use this Hamming-based sparsity as the sparsity objective in all following experiments.

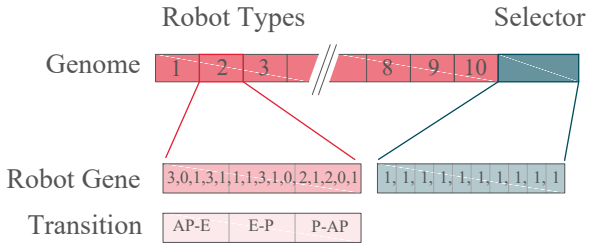


Fig. 4: Two-part genome defining a list of controllers and a selector list to assign robot types to robots (AP: anti-phototaxis, E: exploration, P: phototaxis).

With more than one objective we have a multi-objective optimization problem. We use the standard framework of NSGA-II [15]. In multi-objective optimization we do not get only one best result but all Pareto-optimal solutions. The idea is that we choose our favorite solution from that Pareto set after the algorithm is finished, based on how we want to weight the task performance over sparsity. For the *stay-in* and *stay-out* scenarios we have a third objective of cumulative light.

We configure NSGA-II: tournament selection size four, single-point crossover, mutation as bit-flip with probability 0.04 per bit, population size 100, and 500 or 1,000 generations. We do 20 independent evolutionary runs per setting. Evaluations are run in simulation for 2,000 ticks or 6,000 ticks.

We use the so-called hypervolume to easily visualize the performance of our approach. The hypervolume is the volume of the space that is Pareto-dominated by current solutions and bounded by a reference point. Reference point is the nadir point (worst case, both objectives minimal) [16], here: $[0, 0]$.

D. Swarm configurations as genome

For the genetic algorithm of NSGA-II we need to define a genome, that is, an encoded representation of swarm configurations C. We could enumerate all 750^3 controllers and then define a genome that contains only $N = 10$ as assignment of which robot uses which robot type. However, a mutation of such a number may switch to a very different behavior and it would be particularly different to achieve sparsity. Instead we introduce a two-part genome (see Fig. 4): ten full parameter sets of controllers ('robot type') and ten numbers to assign robots to this limited list of ten controllers ('selector'). Each controller ('robot gene') consists of the 3×5 transition parameters. This genome definition is complete in the sense of not excluding potentially usefully solutions. A fully heterogeneous swarm with $M = N = 10$ is possible if all numbers in the selector are different. A homogeneous swarm with $M = 1$ is possible if all numbers in the selector are the same. Hence, a homogeneous solution may still be relatively unlikely to achieve by chance but there are only

10^{10} possibilities instead of $(750^3)^{10}$. Also each of the 750^3 robot types has a non-zero chance to be part of the solution.

IV. Results

In a preliminary test we investigate the impact of the selector part in the genome (Fig. 4). A comparison of genomes with selectors versus genomes without selectors is given in Fig. 5c that shows the hypervolume over generations. The difference is significant and in favor of the genome with selector. A typical example of a successful run is shown in Fig. 5a (initial setup) and Fig. 5b (final situation).

A. Basic scenario

For the basic scenario we run NSGA-II for 500 generations. Evaluations in the simulation are run for 2,000 ticks. The results are given in Fig. 6, top row. The hypervolume plot (Fig. 6a) shows a steep initial increase as expected and saturates after about 200 generations. Fig. 6b shows the Pareto front of all 20 independent evolutionary runs. In average a coverage of about 47% is reached. Hence, the task is successfully solved. Several solutions, including the best solution, have a sparsity of one (homogeneous swarm). Other solutions have sparsity of about 0.85 corresponding to swarms with homogeneous subpopulation of nine robots and one different robot. The rectangular shape of the Pareto front indicates a simple multi-objective problem, here probably because sparsity is easily achieved.

The evolved best behavior[1] is homogeneous and performs as expected. Robots explore the area until they find a cylinder, then switch to phototaxis and drag the cylinder to the target zone. After dropping the cylinder, they move away from the target with anti-phototaxis and explore the arena again. We investigate the evolved controllers and find that two aspects of the behavior need to be fine-tuned for good performance: when to drop the cylinder and for how long to explore. All successful behaviors evolved the same parameters for these two aspects. Solutions with heterogeneous swarms have an arguably dysfunctional robot that always stays in *explore* or *anti-phototaxis* state. This way the robot takes itself out of the game, such that it is not helping in transporting cylinders but it also does not interfere with other robots. Hence, one could argue that the swarm regulates the swarm density this way. Still, a heterogeneous swarm seems suboptimal here.

B. Stay-in scenario

For the stay-in scenario we run NSGA-II for 1,000 generations. Evaluations in the simulation are run for 2,000 ticks. In addition to the objectives coverage and sparsity, we have a third objective of staying close to the light source. The hypervolume plot in Fig. 6c shows that the evolutionary approach is effective again. Here the Pareto plot (Fig. 6d) represents three objectives with sparsity being color-coded. The objectives of staying close to the light source and achieving high coverage are in conflict. This can be seen also in the Pareto plot (Fig. 6d), where the area around the upper right corner is empty. Instead, the solutions spread along a diagonal to balance the two objectives. Also a gradient in sparsity can be seen. The closer solutions get to the upper right corner the more they tend to be heterogeneous. These heterogeneous swarms use specialized robots that stay at the light (called stay-bot) and other robots that move cylinders (called move-bot). Homogeneous swarms use a generalist strategy where robots move cylinders while spending as much time at the light as possible. The Pareto plot shows that heterogeneous swarms tend to perform better.

We take a look at the behavior of a swarm with eight move-bots and two stay-bots.[2] Move-bots show a behavior comparable to the behavior in the basic scenario. However, they spend less time in *exploration*. They move away from the light source but then try to almost immediately return again while grabbing any cylinder they encounter. Seemingly this is a good compromise between staying close to the light and still doing some construction. This behavior is also optimized for the length of the evaluation because in longer runs the short exploration time would not often enough be sufficient to find cylinders. The cylinders that were closer to the light initially would then already be used for construction.

Stay-bots remain in *phototaxis* and always stay near the light source. Interestingly, they take a cylinder with them. At the beginning of the simulation stay-bots explore the arena until they find a cylinder. Only then they switch to phototaxis and move towards the light source with the cylinder. A stay-bot then ensures to stay in *phototaxis* in the following way. The transition from *phototaxis* to *anti-phototaxis* is only triggered if there is no cylinder in range. By bringing a cylinder, the stay-bot makes sure to stay at the light. It is unclear whether this is the best behavior to ensure staying in *phototaxis*. Other options could be to prevent the transition by setting impossible light parameters.

C. Stay-out scenario

For the *stay-out* scenario we run NSGA-II for 1,000 generations. Evaluations in the simulation are run for 6,000 ticks to avoid behaviors being optimized for short evaluations.

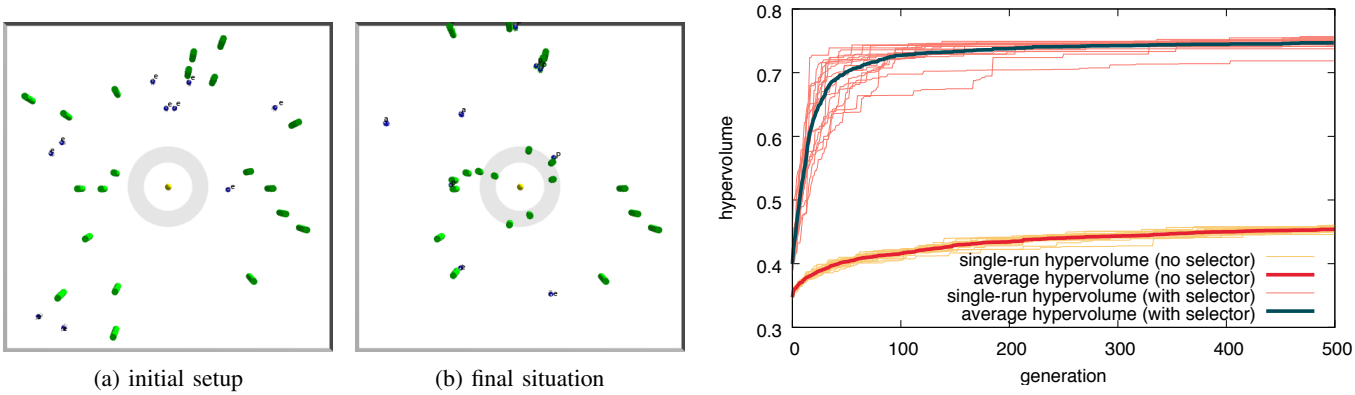
The hypervolume plot (Fig. 6e) indicates effective evolutioanry runs and shows a number of bigger steps even after 600 generations. The Pareto plot (Fig. 6d) shows a clear front. Low sparsity seems to be achieved easily.

We take a look at a particular solution[3]. The swarm has nine robots of one type and only one robot with a different controller that stays in exploration state at all time. All other robots grab a cylinder and enter *phototaxis*. They leave *phototaxis* if they have stayed minimal time $\theta_T = 2$ [time steps] in this state and sense another cylinder $\theta_D = 1$. They drop the cylinder, start *anti-phototaxis* again, and stay for time $\theta_T = 3$ in that state. Then *exploration* follows and the robot grabs a cylinder again. This behavior leads to a fast

[1] video online: http://zenodo.org/record/1258018/files/1_Basic.mp4
(displayed letters indicate e: exploration, p: phototaxis, and a: anti-phototaxis)

[2] video online: https://zenodo.org/record/1258018/files/4_Light.mp4

[3] video online: https://zenodo.org/record/1258018/files/5_Antilight.mp4



(a) initial setup (b) final situation (c) hypervolume

Fig. 5: Left and middle panel: screenshots of the simulation, initial setup with randomly distributed robots (blue) and cylinders (green) and final situation with six cylinders in the target zone. The swarm was evolved for the basic scenario. Right panel: hypervolume plot of evolutionary runs with and without selector in the genome, with selector clearly better.

delivery of cylinders from areas far away from the target zone. Cylinders are moved rapidly from the edges of the arena to the target zone but not necessarily all the way. The final delivery to the target zone is done by robots of the same type but can be considered a different behavior. If there are many cylinders near a robot, it cycles through the states quickly. It drags and drops a cylinder bit by bit to the target zone. As there is usually a group of cylinders, it drags a different cylinder in each iteration, hence, moving the whole cluster. As the light is blocked by cylinders, robots can hide in the shadow of cylinder clusters and satisfy the *stay-out* objective.

V. Discussion and Conclusion

We have successfully applied the population coding paradigm to swarm construction. The results indicate that the emergence of heterogeneous and homogeneous swarms is not always intuitive and hence indicates a possibly challenging problem. Whether heterogeneous swarms can be efficient for a given scenario depends on a division of labor, a subsequent task allocation, and task specialization (cf. [7]). Whether division of labor is possible and useful depends on whether the task allows for a certain modularity in the form of subtasks (e.g., heterogeneous solution for the *stay-in* scenario with stay-bots and move-bots). These subtaks then need to be worked on by a minimum number of assigned robots each. We would expect each subtask to require a different behavior, hence allowing for and possibly requiring task specialization to outperform competing homogeneous swarms (e.g., advantage of having stay-bots that stay close to the light while move-bots do the construction). Additional complexity is introduced if the subtasks are mutually dependent such that a subpopulation specialized on task A may be of no use without a subpopualation specialized on task B and vice versa (not the case for the stay-bot and move-bots example). The required evolutionary dynamics corresponds to the problem of evolving a communication system with the chicken-and-egg problem that senders without receivers are of no use and vice versa.

Concerning the actual construction task we found that relatively simple behaviors are effective. For example, in the discussed construction behaviors there is no specific strategy to place cylinders. However, the robots tend to place them efficiently because they place them in bright areas. The light sensor detects low light intensities behind already placed cylinders and prevents the robot from placing cylinders in inefficient positions. This can be interpreted as a form of stigmergy but is obviously dependent on the specific scenario and light setting here.

In future work we plan to increase the task complexity, to add a further subtask, and to study self-repair mechanisms to react to damage in the construction.

References

[1] J. C. Bongard, "Evolutionary robotics," *Communications of the ACM*, vol. 56, no. 8, pp. 74–83, 2013. [Online]. Available: http://doi.acm.org/10.1145/2493883

[2] L. Panait and S. Luke, "Cooperative multi-agent learning: The state of the art," *Autonomous Agents and Multi-Agent Systems*, vol. 11, no. 3, pp. 387–434, 2005.

[3] H. Hamann, *Swarm Robotics: A Formal Approach*. Springer, 2018.

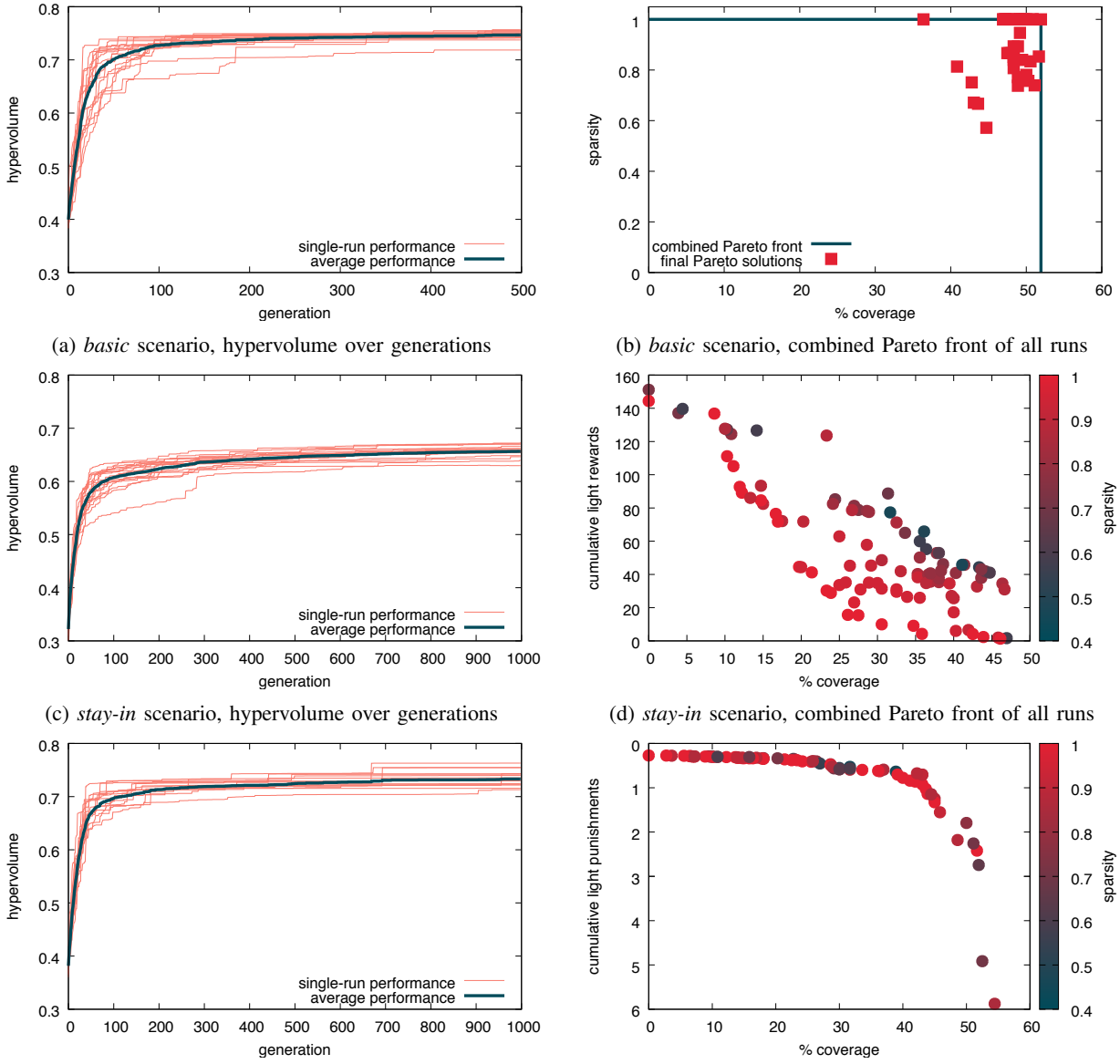
[4] V. Trianni, *Evolutionary Swarm Robotics - Evolving Self-Organising Behaviours in Groups of Autonomous Robots*, ser. Studies in Computational Intelligence. Berlin, Germany: Springer, 2008, vol. 108.

[5] H. Hamann, G. Valentini, and M. Dorigo, "Population coding: A new design paradigm for embodied distributed systems," in *10th Int. Conf. on Swarm Intelligence, ANTS 2016*, ser. Lecture Notes in Computer Science. Springer, 2016, pp. 173–184. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-44427-7_15

[6] J. Gomes, P. Mariano, and A. L. Christensen, "Cooperative coevolution of partially heterogeneous multiagent systems," in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2015, pp. 297–305.

[7] E. Ferrante, A. E. Turgut, E. Duéñez-Guzmàn, M. Dorigo, and T. Wenseleers, "Evolution of self-organized task specialization in robot swarms," *PLoS Comput Biol*, vol. 11, no. 8, p. e1004273, 08 2015. [Online]. Available: http://dx.doi.org/10.1371%2Fjournal.pcbi.1004273

[8] A. Prorok, M. A. Hsieh, and V. Kumar, "Formalizing the impact of diversity on performance in a heterogeneous swarm of robots," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 5364–5371.



(a) *basic* scenario, hypervolume over generations

(b) *basic* scenario, combined Pareto front of all runs

(c) *stay-in* scenario, hypervolume over generations

(d) *stay-in* scenario, combined Pareto front of all runs

(e) *stay-out* scenario, hypervolume over generations

(f) *stay-out* scenario, combined Pareto front of all runs

Fig. 6: Results of scenarios *basic* (top row), *stay-in* (middle row) and *stay-out* (bottom row). Hypervolume over generations (left column) and combined Pareto front of all runs (right column), 20 independent evolutionary runs.

[9] D. Kengyel, H. Hamann, P. Zahadat, G. Radspieler, F. Wotawa, and T. Schmickl, "Potential of heterogeneity in collective behaviors: A case study on heterogeneous swarms," in *PRIMA 2015: Principles and Practice of Multi-Agent Systems*, ser. Lecture Notes in Computer Science, Q. Chen, P. Torroni, S. Villata, J. Hsu, and A. Omicini, Eds. Springer, 2015, vol. 9387, pp. 201–217.

[10] F. Ducatelle, G. A. Di Caro, and L. M. Gambardella, "Cooperative self-organization in a heterogeneous swarm robotic system," in *Proceedings of the 12th Conf. on Genetic and evolutionary computation (GECCO)*. ACM, 2010, pp. 87–94.

[11] H. Hamann, G. Valentini, Y. Khaluf, and M. Dorigo, "Derivation of a micro-macro link for collective decision-making systems: Uncover network features based on drift measurements," in *13th International Conference on Parallel Problem Solving from Nature (PPSN 2014)*, ser. Lecture Notes in Computer Science, T. Bartz-Beielstein, Ed. Springer, 2014, vol. 8672, pp. 181–190. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10762-2_18

[12] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. Caro, F. Ducatelle, M. Birattari, L. M. Gambardella, and M. Dorigo, "ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems," *Swarm Intelligence*, vol. 6, no. 4, pp. 271–295, 2012. [Online]. Available: http://dx.doi.org/10.1007/s11721-012-0072-5

[13] J. H. Holland, *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: Univ. Michigan Press, 1975.

[14] N. Hurley and S. Rickard, "Comparing measures of sparsity," *IEEE Transactions on Information Theory*, vol. 55, no. 10, pp. 4723–4741, 2009.

[15] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *International Conference on Parallel Problem Solving From Nature*. Springer, 2000, pp. 849–858.

[16] D. Hadka, "Beginner's guide to the MOEA framework," 2016.