2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

A Distributed Strategy for Gait Adaptation in Modular Robots

David Johan Christensen, Ulrik Pagh Schultz and Kasper Stoy
The Maersk Mc-Kinney Moller Institute
University of Southern Denmark
{david, ups, kaspers}@mmmi.sdu.dk

Abstract— **In this paper we study online gait optimization for modular robots. The learning strategy we apply is distributed, independent on robot morphology, and easy to implement. First we demonstrate how the strategy allows an ATRON robot to adapt to faults and changes in its morphology and we study the strategy's scalability. Second we extend the strategy to learn the parameters of gait-tables for ATRON and M-TRAN robots. We conclude that the presented strategy is effective for online learning of gaits for most types of modular robots and that learning can effectively be distributed by having independent processes learning in parallel.**

I. INTRODUCTION

Modular robots are often designed with a distributed and flexible morphology. The modules may be combined in various ways to construct different robot configuration and each module is controlled by it own local microcontroller. These design characteristics entails that conventional centralized control strategies are not always a convenient choice for modular robots.

Instead we study distributed control strategies which generally may have several advantages over centralized strategies, e.g., in terms of ease of implementation, robustness, reconfigurability, scalability, biological plausibility, etc. Specifically, for the purpose of online locomotion learning, we study distributed adaptive strategies where each module itself optimizes its own behavior based on its local context and local interactions and thereby indirectly optimizes the global behavior of the robot. We hypothesize that such strategies may be more robust and flexible since it may be indifferent to the robot's morphology and can online adapt to module failures or morphology changes. Ultimately, we anticipate that by studying such distributed strategies we may gain insights into how adaptive sensory-motor coordination can emerge and self-organize from billions of individual cells in biological organisms.

In previous work we proposed and studied a distributed learning strategy appropriate for gaits optimization in modular robots [2]. The strategy was simple to implement and independent on the robot's specific morphology. We validated the strategy both in simulation and on physical ATRON robots and found that the strategy was sufficient to learn quite efficient locomotion gaits for a large range of different morphologies up to 12-module robots. A typical learning trial converged in less than 15 minutes depending on the size and type of the robot. In this paper we give additional experimental validation in simulation and propose an extension to this previous published strategy. In Section III we summarize the basic strategy and extend it to optimization of gait-tables, which makes it applicable to most modular robots. The rest of the paper presents simulated experiments. First with the basic strategy to study its robustness and scalability and then with the extended strategy for online optimization of gait-table controlled ATRON and M-TRAN robots. Fig. 1 illustrates some of the simulated robots we use for experiments in this paper.

II. RELATED WORK

Here, we review related work on evolutionary adaptation and online learning of modular robots for the task of locomotion. Karl Sims pioneered the field in the early 90's by co-evolving the morphology and control of simulated modular robots [12]. Later works succeeded in transferring the co-evolved robots from simulation to hardware [6], [8]. An example of adaptation by evolution in modular robots was conducted by Kamimura et al., who evolved the coupling parameters of central pattern generators for straight line locomotion of M-TRAN self-reconfigurable robots [4]. Although appealing, one challenge with evolutionary approaches is to bridge the reality gap and once transferred the robot is typically no longer able to adapt. To solve this limitaiton optimization of locomotion gaits can be performed online. This was studied by Marbach and Ijspeert on the YaMoR modular robotic system [9]. Their strategy was based on Powell's method, which performed a localized search in the space of selected parameters of central pattern generators. Parameters were manually extracted from the modular robot by exploiting symmetries. Follow-up work by Sproewitz et al. demonstrated online optimization of 7 parameters on a physical robot in roughly 15 minutes [14]. As is the case in our paper, they try to realize simple, robust, fast, model-less, life-long learning on a modular robot. The main difference is that we seek to automate the controller design completely in the sense that no parameters have to be extracted from symmetric properties of the robot. Further, our approach utilizes a form of distributed reinforcement learning. A similar approach was taken by Maes and Brooks who performed distributed learning of locomotion on a 6-legged robot [7]. The learning was distributed to the legs themselves. Sim-

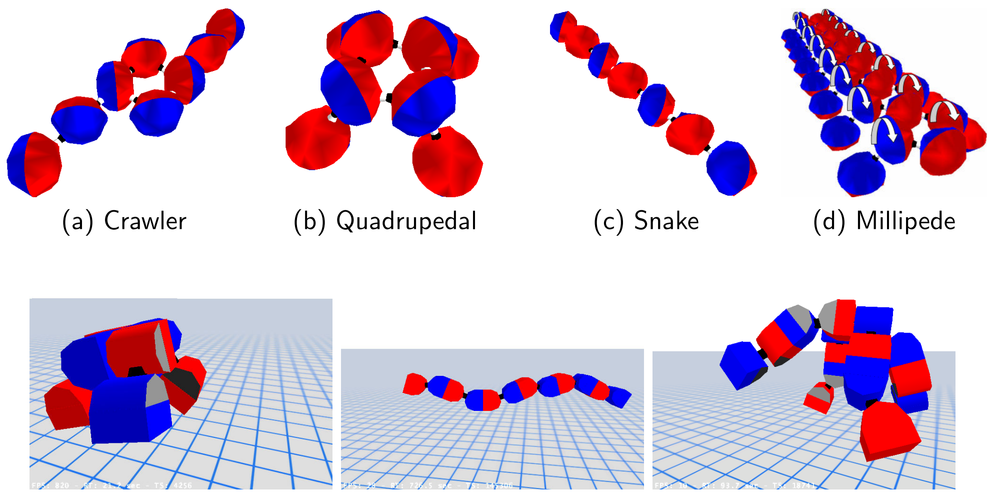


Fig. 1. Various simulated ATRON and M-TRAN robots used in experiments.

ilarly, in the context of multi-robot systems, distributed reinforcement learning has been applied for learning various collective behaviors [10]. Our strategy is independent on the robot's specific morphology. Similarly, Bongard et al. demonstrated learning of locomotion and adaptation to changes in the configuration of a modular robot [1]. They took a self-modeling approach, where the robot developed a model of its own configuration by performing basic motor actions. In a physical simulator a model of the robot configuration was evolved to match the sampled sensor data (from accelerometers). By co-evolving the model with a locomotion gait, the robot could then learn to move with different morphologies. Our work presented here is similar in purpose but different in approach: Out strategy is simple, model-less and computational cheap to allow implementation on the small embedded devices that modular robots usually are.

III. Adaptation Strategy

A. Basic Learning Strategy

We utilize a simple stateless reinforcement learning strategy where each module in the robot is controlled by Algorithm 1. Initially each module executes all possible actions, A, in random order and initializes its action value estimation, $Q[A]$, with the rewards received. After this initialization phase, in a learning iteration, every module will perform an action and then receive a global reward for that learning iteration. Each module estimates the value of each of its actions with an exponential moving average, which suppress noise and ensures that if the value of an action changes with time so will its estimation. The algorithm can be categorized as a $TD(0)$ with discount factor $\gamma = 0$ and with no representation of the sensor state [15].

A module can perform a small fixed number of actions. Each module independently selects which action to perform based on a ϵ-greedy selection policy, where a module

Algorithm 1 Basic Learning Strategy.

```
/*
 * $Q[A]$ is expected reward $R$ of choosing Action $A$.
 * $\alpha$ is a smoothing factor
 * $1-\epsilon$ is the proportion of "greedy" action selections.
 */
Initialize $Q[A] = R$, for all $A$ evaluated in random order
loop
    if $max(Q) < R$ then
        Repeat Action $A$
    else
        Select Action $A$ with max $Q[A]$ with prob. $1-\epsilon$, otherwise
        random action
    end if
    Execute Action $A$ for $T$ seconds
    Receive Reward $R$
    Update $Q[A] += \alpha \cdot (R - Q[A])$
end loop
```

selects the action with highest estimated reward with a probability of $1 - \epsilon$ and a random action otherwise.

Performance of a module is highly coupled with the behavior of the other modules in the robot. Therefore, the best action of a module is non-stationary. It can change over time when other modules change their action. Hence, the learning speed is limited by the fact that it must rely on randomness to select a fitter but underestimated action a sufficient number of times before the reward estimation becomes accurate. To speedup the estimation of an underestimated action we use a heuristics to accelerate the learning: If the received reward after a learning period is higher than the highest estimation of any action, the evaluated action may be underestimated and fitter than the current highest estimated action. Note that this is not always true since the fitness evaluation may be noisy. Therefore, a simple heuristic is to repeat the potentially underestimated action, to accelerate the estimation accuracy and presumably accelerate the learning, see Algorithm 1. In our previous work we found that this heuristic could significantly increase the convergence

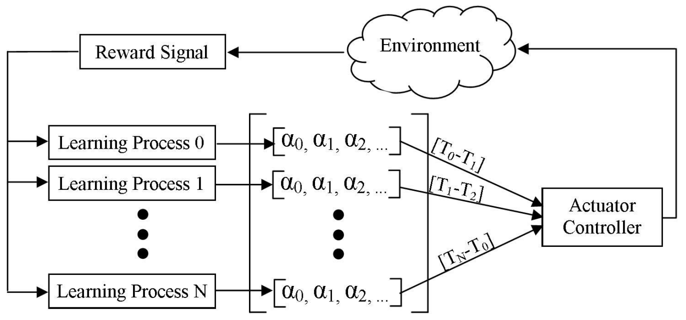


Fig. 2. Illustration of gait-table based learning strategy for a single module. Independent and parallel learning processes learn each entry in the gait-table. Therefore, each module learns its own column of the gait-table.

speed [2].

B. *Learning Strategy Extended to Gait Tables*

In the basic learning strategy each module learns to always perform a single action, e.g. rotate clockwise. To enable a more versatile learning strategy which may work on any module type we combine the strategy with gait control tables. Originally proposed by Mark Yim [16], gait-tables contains set-points for the actuator where the columns represent actuators and the rows time intervals. To learn the set-points in a gait-table we let each module learn the set-points of its own columns in the gait-table. Each module runs one parallel learning process, i.e. Algorithm 1, per entity in its columns. Each learning process selects a set-point amongst a set of predefined set-points. The learning processes learn independently and in parallel based on a shared reward signal. This extended strategy is illustrated in Fig. 2. To utilize this approach for a given system we must define the set of set-points that can fill the table and the number of rows in the table, the number of columns is indirectly automatically adjusted when adding or removing modules.

IV. EXPERIMENTAL SETUP

A. *Physics-based Simulation of Modules*

Simulation experiments are performed in an open-source simulator named Unified Simulator for Self-Reconfigurable Robots (USSR) [3]. We have developed USSR as an extendable physics simulator for modular robots. Therefore, USSR includes implementations of several existing modular robots, e.g., ATRON and M-TRAN as utilized in this paper. The simulator is based on Open Dynamics Engine [13] which provides simulation of collisions and rigid body dynamics. Through socket connections USSR is able to run module controllers which can also be cross-compiled for the physical platform.

The ATRON module [11] is comprised of two hemispheres that can rotate infinite relative to each other. On each hemisphere a module has two passive female and two actuated male connectors. The parameters of the simulation model, e.g. strength, speed, weight, etc., has been calibrated to match the existing hardware platform to ease the transfer of controllers developed in simulation to the physical modules.

Exp.	Robot	α	$1-\epsilon$	T
SR & Faults	ATRON	0.1	0.8	7
Scalability	ATRON	0.1	0.8	7
Gait-Table	ATRON	0.1	0.96	7
Gait-Table	M-TRAN	0.0333	0.96	1.5

TABLE I
LEARNING PARAMETERS

The M-TRAN module fills two cells in a cubic lattice and has six connector surfaces. Each module have two actuators which can rotate in the interval $\pm 90^o$. We have implemented a model of the M-TRAN III module in the USSR simulator based on available specifications [5]. However, we do have access to the physical M-TRAN modules. Therefore, although the kinematics is correct, specific characteristics might be somewhat different from the real system. We accept this, since our purpose is to validate the gait-table based learning strategy on a different system, not to find efficient locomotion gaits for M-TRAN robots.

B. *Learning Parameters and Reward Signal*

In the following four experiments each module runs identical learning controllers with parameters set as indicated in Table I. In some experiments we compare with randomly moving robots, i.e. we set $1 - \epsilon = 0$. Each module in a robot share and optimize its behavior based on the same reward signal. The reward is computed as the distance traveled by the robots center of mass in the duration of a learning iteration:

$$Reward = Distance\ traveled\ by\ robot\ in\ T\ seconds$$

The choice of T is selected to match the actuation speed of the module type, α balances the amount of noise in the reward signal against convergence time, and ϵ controls the exploration/exploitation tradeoff as a function of the number of parallel learning processes.

C. *Action Space*

The basic and extended learning strategy required us to define a set of actions/set-points. We select actions/set-points that are module type specific but somewhat generic with respect to robot morphology.

The two experiments with the basic strategy utilize ATRON modules that always perform one of the following three actions: *HomeStop* (rotates to 0 degrees and stop), *RightRotate* (rotate clockwise 360 degrees) and *LeftRotate* (rotate counterclockwise 360 degrees). After a learning iteration, a module should ideally be back at its home position to ensure repeatability. Therefore, the modules will synchronize their progress to follow the rhythm of the learning iteration.

In the gait-table experiments, to study how the strategy works on a system with a different kinematics, we will use joint limited ATRON modules, i.e., which cannot be

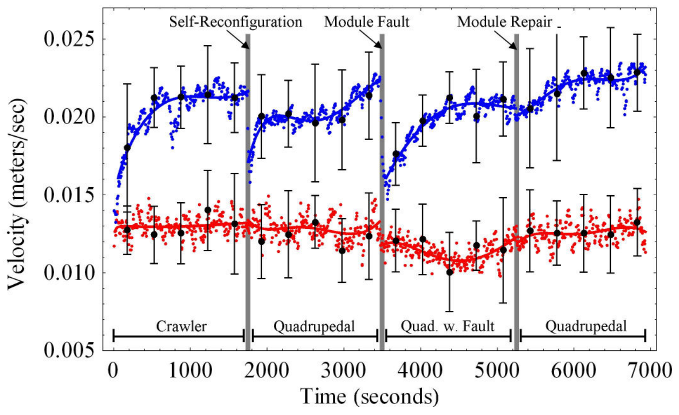


Fig. 3. Adaptation of gait after self-reconfiguration and module fault. Initially the robot is of a crawler type, it then self-reconfigures to a quadrupedal, then a module fails and finally the module again becomes functional. In each new case, the learning enables the robot to adapt to the new situation, by changing the locomotion gait. The graph is the average of 10 trials, with standard deviation as error bars. The bottom line is 10 trials of the equivalent robot moving randomly.

rotated infinitely but is limited to a ±90 degree interval. In effect, the gait-table based learning strategy must find alternative gaits to move the robots, instead of gaits based on continuous rotations. The gait-table has five rows, so each module must learn five angle values from the set-point set: {-60, -30, 0, 30, 60} degrees. The M-TRAN module has two actuators. Therefore we let each actuator be controlled by independent gait-tables. Each gait-table has five rows, where an entry can contain one of three set-points: {-30, 0, 30} degrees.

Further, for each robot morphology we define an initial pose that the actuation is performed relative to. Selecting a pose is a tradeoff between high potential to move and being stable so that the robot does not fall over while learning.

V. EXPERIMENTS

A. *Self-Reconfiguration and Faults*

In this experiment, we study the basic learning strategy's ability to adapt to changes in robot morphology. Initially we let a crawler type robot (8 modules) learn to move, see Fig. 1(a). At learning iteration 250 (after 29 minutes), the robot is then programmed to self-reconfigure into a quadrupedal type robot, see Fig. 1(b). Afterwards the learning is continued without resetting the learning system. After additional 250 iterations, we simulate a module failure by stopping a leg module in a non-home position. 250 iterations later we reactivate the module and let the learning continue for another 250 iterations.

Fig. 3, shows the average results of 10 trials. After both the self-reconfiguration and module fault, we observe a drop in fitness as expected. In both cases, the learning system is able to adapt to its changed morphology and regain a higher velocity. In the case there a leg module is reactivated there is no initial drop in fitness, but afterwards the robot learns again to use its leg and the average velocity increases again.

B. *Scalability*

To study the scalability of the learning strategy we performed experiments with a scalable robot. We utilized a millipede robot as shown in Fig. 1(d). This robot has a best-known controller as indicated in the figure. In the following experiments, we vary the number of legs from 4 to 36 in steps of 4 with 10 learning trials per robot.

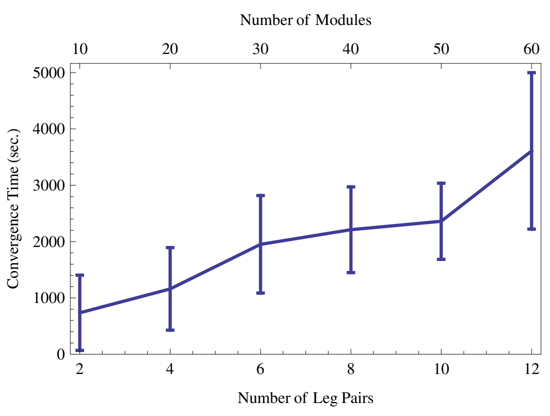
We define the time of convergence as the time at which 85 % of the leg modules has learned to contribute to the robot movement. That is, the leg module rotates either left or right dependent on its position in the robot and the direction of locomotion. The time to converge is shown in Fig. 4(a). As expected, an increase in the number of modules also increase the convergence time, the relation is approximately linear for this robot in the interval shown. The increase in convergence time is rather slow, for each module added the convergence time is prolonged with 52 seconds (based on a least square fit: $convergenceTime = 52 \cdot \#modules + 182$ seconds). Beyond this interval of up to 60 modules, divergence becomes the dominating factor, i.e. the robot forgets already learned behavior.

We measure learning divergence as a major drop in number of leg modules contributing to moving the millipede. The frequency of diverges of each robot is shown in Fig. 4(b). We observe that the divergence frequency increases with the number of modules. The reason behind this is that as the number of modules increase the effect that any individual module has on the robot decreases. Therefore, for a given module the estimates for each of its actions will almost be identical and small disturbance can cause the divergence effect.

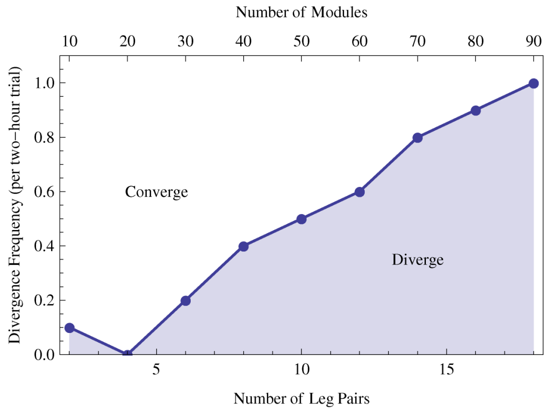
C. *Gait-Table Learning with ATRON*

In this experiment we use the extended learning strategy to optimize gait control tables for a snake (chain with seven modules), millipede-2 (two leg pairs, 10 modules), millipede-3 (three leg pairs, 15 modules) and a quadrupedal (8 modules).

Fig. 5 shows the convergence as the average velocity achieved over time compared with randomly moving robots. Only the results for two of the robots are shown but they are typical. Note, that a robot tends to quickly learn a better than random gait, and that this gait gradually improve over time. We, observe that the learning strategy is able to find significantly better than random gaits for the different robots. Compared to blind random search optimization the convergence speed is similar but the learning strategy finds significantly better gaits, e.g., on average 9.9 cm/sec and 13.3 cm/sec respectively for the millipede-3 robot. Although, the robots moves slower than if they could perform unlimited rotation, the gaits found are quite efficient. Also, note that in the case of the snake robot the basic learning strategy fails

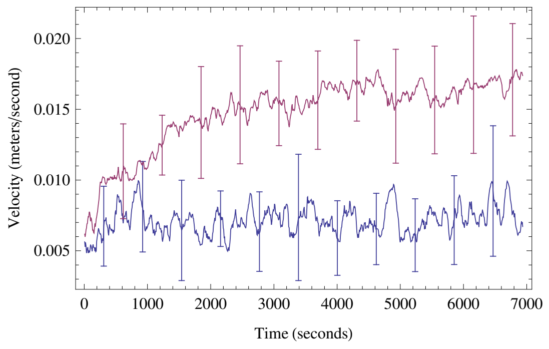


(a) Convergence

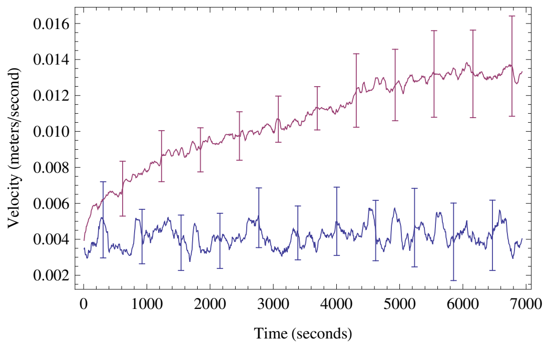


(b) Divergence

Fig. 4. (a) Convergence time versus number of modules. (b) Divergence versus number of modules. The robots are millipedes with 4 to 24 legs. Error bars indicate one standard deviation.



(a) Snake



(b) Millipede-3

Fig. 5. Convergence graphs for the two different robots assembled from joint-limited ATRON modules. For comparison the average velocity of random moving robots is also shown. Each graph is the average of 10 trials. Errorbars indicate standard deviation.

to converge since the robot entangles itself, while this extended strategy converges to undulation style gaits. All the 40 experimental trials converged to good performing gaits. Divergence happens in a few cases when a snake robot rolls upside down during learning and then had to learn to move from this state.

Some typical gaits found: The snake is moving with a side-winding gait, with a traveling wave down the chain. The snake lifts parts of its body off the ground as it moves. A typical quadrupedal gait could use a back foot partly as a wheel, partly as a foot. Its side legs moves back and forward for movement, while the front leg is used just for support. The millipede-2 has a trot style gait, where the diagonal opposite legs move together. The millipede-3 uses a similar gait with each leg oscillating back and forward with some unrecognizable scheme of synchronization between the legs.

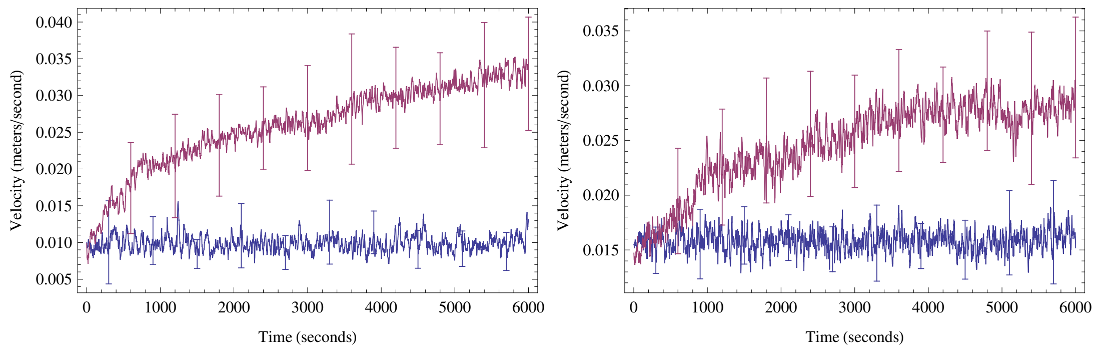
D. *Gait-Table Learning with M-TRAN*

In this experiment we apply the gait-table based learning strategy on three simulated M-TRAN robots: A 6-module caterpillar (12 DOF), a 4-module mini walker (8 DOF) and an 8-module walker (16 DOF).

Fig. 6 shows convergence graphs for two of the robots. Notice, that the performance of the gaits quickly becomes better than random and that the gaits gradually improves over time. The learning succeeds in finding efficient gaits for all three robots. Because of the short learning iteration (T=1.5 seconds) even a pose shift can be measured as quite high velocity, why randomly moving robots incorrectly seems to move quite fast. We observe that the large learning space leaves room for incremental learning.

A major challenge with learning M-TRAN gaits is that the robot often falls over while learning. This happened in 23 percent, 8 percent and 47 percent of the two hour trials with the mini walker, caterpillar and walker respectively. These trials were censored away in the presented results, which is based on 10 completed trials per robot.

Some typical learned gaits: Typical gaits for the mini walker consist of hopping movement, with two modules producing movement and two modules creating stability. For the caterpillar, the learning typically finds gaits either with a horizontal traveling wave down the chain of modules or gaits that uses the head and tail modules to push on the ground. Successful gaits for the walker take relative short steps, since the robot would otherwise fall over. In one example trial the walker use three legs to produce movement, while the forth leg is kept lifted off the ground in front of the robot.



(a) Caterpillar (b) Walker

Fig. 6. Average velocity as a function of time for two M-TRAN robots. Each graph is the average velocity of 10 independent trials. Average velocity of randomly moving robots is shown for comparison. Errorbars indicate one standard deviation.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we presented simulated experiments on distributed adaptation of locomotion for a range of robots constructed from ATRON and M-TRAN modules. For our previously published basic strategy we presented simulated experiments with faults and self-reconfiguration that illustrated the advantages of utilizing a distributed and configuration independent learning strategy. We saw that the modules after reconfiguration were able to learn to move with a new morphology and adapt to module faults. In simulation, we studied the scalability characteristics of the learning strategy and found that it could learn to move an robot with up to 60 modules (60 DOF millipede). However, the effects of divergence in the learning would eventually become dominant and prevent the robot from being scaled further up. We also found that the convergence time increased slowly approximately linear, with the number of modules within the functional range.

We extended the basic learning strategy to learn the set-points in a gait-table. This extended strategy is distributed, with several parallel and independently learning processes running on each module. We experimentally validated that the extended strategy was able to learn effective gaits for the different robots and module types. However, as anticipated, the increased size of the learning space came at the cost of prolonged time to learn a gait. Yet, even the most complex gaits are typically learned within one hour. In conclusion, learning can effectively be distributed by introducing independent processes learning in parallel. Further, the extended learning strategy based on gait-tables is a simple to implement strategy, which can be used on almost any existing modular robotic platform.

Future work will improve our distributed approach by optimizing floating point parameters of central pattern generators instead of discrete action or set-points in gait-tables. Further, we will replace the exponential moving average with an more efficient stochastic gradient hill climbing strategy.

References

[1] J. Bongard, V. Zykov, and H. Lipson. Resilient machines through continuous self-modeling. *Science*, 314(5802):1118–1121, 2006.

[2] D. J. Christensen, M. Bordignon, U. P. Schultz, D. Shaikh, and K. Stoy. Morphology independent learning in modular robots. In *Proceedings of International Symposium on Distributed Autonomous Robotic Systems 8 (DARS 2008)*, pages 379–391, 2008.

[3] D. J. Christensen, U. P. Schultz, D. Brandt, and K. Stoy. A unified simulator for self-reconfigurable robots. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008.

[4] A. Kamimura, H.Kurokawa, E. Yoshida, S. Murata, K. Tomita, and S. Kokaji. Automatic locomotion design and experiments for a modular robotic system. *IEEE/ASME Transactions on Mechatronics*, 10(3):314–325, June 2005.

[5] H. Kurokawa, K. Tomita, A. Kamimura, S. Kokaji, T. Hasuo, and S. Murata. Distributed self-reconfiguration of M-TRAN III modular robotic system. *International Journal of Robotics Research*, 27(3-4):373–386, 2008.

[6] H. Lipson and J.B. Pollack. Automatic design and manufacture of robotic lifeforms. *Nature*, 406:974–978, 2000.

[7] P. Maes and R. A. Brooks. Learning to coordinate behaviors. In *National Conference on Artificial Intelligence*, pages 796–802, 1990.

[8] D. Marbach and A.J. Ijspeert. Co-evolution of configuration and control for homogenous modular robots. In *Proc., 8th Int. Conf. on Intelligent Autonomous Systems*, pages 712–719, Amsterdam, Holland, 2004.

[9] D. Marbach and A.J. Ijspeert. Online Optimization of Modular Robot Locomotion. In *Proceedings of the IEEE Int. Conference on Mechatronics and Automation (ICMA 2005)*, pages 248–253, 2005.

[10] M. J. Mataric. Reinforcement learning in the multi-robot domain. *Auton. Robots*, 4(1):73–83, 1997.

[11] E. H. Østergaard, K. Kassow, R.Beck, and H. H. Lund. Design of the atron lattice-based self-reconfigurable robot. *Auton. Robots*, 21(2):165–183, 2006.

[12] K. Sims. Evolving 3d morphology and behavior by competition. In R. Brooks and P. Maes, editors, *Proc., Artificial Life IV*, pages 28–39. MIT Press, 1994.

[13] R. Smith. Open dynamics engine. www.ode.org, 2005.

[14] A. Sproewitz, R. Moeckel, J. Maye, and A. Ijspeert. Learning to move in modular robots using central pattern generators and online optimization. *Int. J. Rob. Res.*, 27(3-4):423–443, 2008.

[15] R.S. Sutton and A.G. Barto. *Reinforcement Learning - An Introduction*. The MIT Press, 1998.

[16] M. Yim. *Locomotion with a unit-modular reconfigurable robot.* PhD thesis, Department of Mechanical Engineering, Stanford University, 1994.