Efficient Distributed Reinforcement Learning Through Agreement

Paulina Varshavskaya, Leslie Pack Kaelbling and Daniela Rus

Abstract Distributed robotic systems can benefit from automatic controller design and online adaptation by reinforcement learning (RL), but often suffer from the limitations of partial observability. In this paper, we address the twin problems of limited local experience and locally observed but not necessarily telling reward signals encountered in such systems. We combine direct search in policy space with an agreement algorithm to efficiently exchange local rewards and experience among agents. We demonstrate improved learning ability on the locomotion problem for self-reconfiguring modular robots in simulation, and show that a fully distributed implementation can learn good policies just as fast as the centralized implementation. Our results suggest that prior work on centralized RL algorithms for modular robots may be made effective in practice through the application of agreement algorithms. This approach could be fruitful in many cooperative situations, whenever robots need to learn similar behaviors, but have access only to local information.

1 Motivation

Distributed robotics systems, such as teams of vehicles, robotic swarms, or modular robots, where each individual agent has to engage in decision-making under uncertainty and coordinate their activity, present a challenge in controller design. Also, they need to be robust and adaptable to changing environments. Both these challenges can be addressed by the application of reinforcement learning (RL) algorithms. However, often only the most straightforward RL techniques such as Q-learning are applied to groups of robots (e.g., [9, 4]), techniques that rely on the Markov property of the world. Distributed robotic systems, on the other hand, have limitations that violate the Markov assumption: individual agents do not have ac-

Paulina Varshavskaya, Leslie Pack Kaelbling, Daniela Rus
Massachusetts Institute of Technology, Computer Science and Artificial Intelligence Laboratory, Cambridge MA, USA, e-mail: {paulina|lpk|rus}@csail.mit.edu

cess to the full state of the world, but only a partial observational window onto it, as delimited by their sensors. In fact, the process is even nonstationary if several agents learn, i.e., change their behavior, at the same time. This partial observability means that more general RL algorithms need to be used in distributed robotics, such as methods based on belief-propagation or methods based on direct search in policy space (e.g.,[1, 12]). The latter have been applied in particular to self-reconfiguring modular robots (SRMRs); the approach [15] proved fruitful in the case of distributed controller generation with a learning algorithm, when the learning was done centrally by a single "agent" from the experience gathered by individual modules with all their observational limitations. However, the fully distributed implementation of this work, where each module learns its own policy from its own local experience and reward only, has not been immediately successful and has required extra heuristics to work even for relatively small SRMR systems [16].

In fact, the observation that individual agents have access only to their own local experience and reward signal goes against a common assumption in the cooperative distributed RL literature (e.g., [12, 3]) that the reward is a global scalar available unaltered to all agents[1]. Instead, the problem for locally sensing SRMR modules or robotic agents in a group is that the local reward is not necessarily a good measure of the agent's performance. For instance, it is possible that the agent's policy is optimal, but the bad choices made by others prevent it from ever finding itself in a configuration from which it can move.

We address here the twin problems of limited local experience and locally observed but not necessarily telling reward signals through the application of agreement algorithms [2]. We demonstrate below that agreement can be used to exchange both local reward and local experience among agents, and will result in a fully distributed implementation of a direct policy search learning algorithm, which learns good policies just as fast as the centralized implementation[2].

In what follows, we first give a concise background on reinforcement learning by policy search in Sect. 2. We then incorporate agreement algorithms into the learning process in Sect. 3. Section 4 gives experimental results on a simulated distributed system: a self-reconfiguring modular robot. We discuss our contribution in light of related work in Sect. 5 and give directions for future research in Sect. 6.

2 Distributed Policy Search

Suppose a group of robots runs a distributed policy search algorithm in the episodic reinforcement learning framework: at each timestep t each robot i observes the situation inside its local sensory and communications range (observation o_t^i) and chooses an action a_t^i according to its own policy π^i. At the end of the episode, each robot receives a scalar reward signal R^i, which is derived from the robot's local mea-

[1] This assumption is not made in general-payoff fully observable stochastic Markov games [6].

[2] The centralized implementation, which serves as a benchmark, can be viewed as an idealized case where all modules have access to all the experience and rewards of all others.

Algorithm 1 Distributed GAPS (DGAPS) for robot i (observation function o)

initialize parameters $\theta_i \leftarrow$ small random numbers
for each episode **do**
  calculate policy $\pi(\theta_i)$
  initialize observation counts $N \leftarrow 0$ and observation-action counts $C \leftarrow 0$
  for each timestep in episode **do**
    observe o_i and increment $N(o_i)$
    choose a_i from $\pi(o_i, \theta_i)$, increment $C(o_i, a_i)$ and execute a_i
  end for
  get local reward R^i and update θ_i according to
  $\theta_i(o_i, a_i) += \eta R^i (C(o_i, a_i) - \pi(o_i, a_i, \theta_i) N(o_i))$
  update $\pi(\theta_i)$ using Boltzmann's law
end for

surements. This process is formalized as a distributed Partially Observable Markov Decision Process (POMDP).

The basic Gradient Ascent in Policy Space (GAPS) algorithm [12] does hill-climbing to maximize the value (that is, long-term expected reward) of the parameterized policy. The derivation starts with noting that the value of a policy π_θ is $V_\theta = E_\theta[R(h)] = \sum_{h \varepsilon H} R(h) P(h|\theta)$, where θ is the parameter vector defining the policy, R is the reward function, and H is the set of all possible experience histories. If we could calculate the derivative of V_θ with respect to each parameter, it would be possible to do exact gradient ascent on the value by making updates $\Delta\theta_k = \alpha \frac{\partial}{\partial \theta_k} V_\theta$. However, we do not have a model of the world that would give us $P(h|\theta)$ and so we will use stochastic gradient ascent instead. The partial derivative of the value of a policy with respect to one policy parameter is (see [12] for full derivation)

$$\frac{\partial}{\partial \theta_k} V_\theta = \sum_{h \varepsilon H} R(h) \left(P(h|\theta) \sum_{t=1}^{T} \frac{\partial}{\partial \theta_k} \ln \pi_\theta(a_t, o_t) \right). \tag{1}$$

The stochastic gradient ascent algorithm operates by summing up traces of log-policy derivatives at every timestep of each learning episode. Each trace reflects the contribution of a single parameter to the estimated gradient. The makeup of the traces will depend on the policy representation.

The GAPS algorithm was originally derived [12] for a lookup-table representation, where there is a parameter $\theta(o,a)$ for each observation-action pair. The fully distributed version of GAPS (DGAPS) is reproduced here for completeness (Alg. 1). The traces are obtained by counting occurrences of each observation-action pair, and taking the difference between that number and the expected number that comes from the current policy. The policy itself, i.e., the probability of taking an action a_t at time t given the observation o_t and current parameters θ is given by Boltzmann's law:

$$\pi_\theta(a_t, o_t) = P(a_t|o_t, \theta) = \frac{e^{\beta\theta(o_t, a_t)}}{\sum_{a \in A} e^{\beta\theta(o_t, a)}}, \tag{2}$$

where β is the inverse temperature.

This gradient ascent algorithm has some important properties. As with any stochastic hill-climbing method, it can only be relied upon to reach a local optimum in the represented space, and will only converge if the learning rate is reduced over time. Furthermore, distributed GAPS will converge to the same local optimum

as its centralized version (theorem 3.1 in [12]) *provided individual learning robots get the same experience and reward as would the central learner in the centralized case*. Clearly, this condition is not satisfied in any distributed robot system where individuals have only access to their own local observations (which may well be different from anybody else's) and compute rewards based on local information. To counteract that limitation, we introduce agreement algorithms into the learning process.

3 Agreement Algorithms in Policy Search

Agreement (also known as consensus or flocking) algorithms are pervasive in distributed systems research. First introduced by Tsitsiklis et al.[14], they have been employed in many fields including control theory, sensor networks, biological modeling, and economics. They are applicable in both synchronous and partially asynchronous settings.

3.1 Basic Agreement Algorithm

Suppose there are N robots and therefore N processors, where each processor i is updating a variable x_i. For ease of exposition we assume that x_i is a scalar, but the results will hold for vectors as well. Suppose further that each i sends to a subset of others its current value of x_i at certain times. Each module updates its value according to a convex combination of its own and other modules' values:

$$x_i(t+1) = \sum_{j=1}^{N} a_{ij} x_j(\tau_j^i(t)), \qquad \text{if } t \in T^i, \tag{3}$$

$$x_i(t+1) = x_i(t), \qquad \text{otherwise}, \tag{4}$$

where a_{ij} are nonnegative coefficients which sum to 1, T^i is the set of times at which processor i updates its value, and $\tau_j^i(t)$ determines the amount of time by which the value x_j is outdated. If the graph representing the communication network among the processors is connected, and if there is a finite upper bound on communication delays between processors, then the values x_i will exponentially converge to a common intermediate value x such that $x_{min}(0) \leq x \leq x_{max}(0)$ (part of Proposition 3.1 in [2]). The agreed-upon value x will depend on the particulars of the averaging process. It will be equal to the exact arithmetic mean of the initial values x_i if the processors make synchronous, pairwise disjoint, symmetrical updates (under sum invariance [2]):

$$x_i(t+1) = x_j(t+1) = \frac{1}{2}\left(x_i(t) + x_j(t)\right). \tag{5}$$

Agreement algorithms can also be used for gradient-style updates performed by several processors simultaneously in a distributed way [14]. However, GAPS is a *stochastic* gradient ascent algorithm, which means that the updates performed climb the estimated gradient of the policy value function $\nabla\hat{V}$, which may, depending on

Algorithm 2 Synchronous average-based collection of experience. This algorithm may be run at the end of each episode, or the procedure labeled **average** may be used at each time step while gathering experience.

send to all n neighbors $message = \langle R, C_o, C_{oa} \rangle$

for all time steps **do**
  average:
  receive from all n neighbors message $m = \langle R^m, C_o^m, C_{oa}^m \rangle$
  average reward $R \leftarrow \frac{1}{n+1}(R + \sum_{m=1}^{n} R^m)$
  average experience
  $C_o \leftarrow \frac{1}{n+1}(C_o + \sum_{m=1}^{n} C_o^m)$, $C_{oa} \leftarrow \frac{1}{n+1}(C_{oa} + \sum_{m=1}^{n} C_{oa}^m)$
  send to all n neighbors $m' = \langle R, C_o, C_{oa} \rangle$
end for

update: GAPS update rule

the quality of the agents' experience, be very different from ∇V, and may not satisfy the assumptions of gradient optimization with agreement. In addition, there may be correlated noise in a system where agents must maintain physical or wireless connections.

3.2 Agreeing on Common Rewards and Experience

Since agreement is a distributed (weighted) averaging procedure, it is immediately applicable to learning scenarios where the cooperative reward R can be expressed as a (weighted) average of the reward signals received by individual learning agents. We expect the agreed-upon $\hat{R}$ to be a more accurate measure of how well the entire system is doing than the individual local reward R^i, which will increase the likelihood of better updates by all robots. The estimate will be exactly right: $\hat{R} = R$ if the robots use synchronous, symmetrical, pairwise disjoint updates (Eq. 5), and the original R is the arithmetic mean of local R^i's, such as when SRMR modules learn a locomotion policy [16].

A theorem in [12] states that a distributed version of GAPS, where each agent maintains and updates its own policy parameters, will make exactly the same updates as the centralized version of GAPS, provided the same experience and the same rewards. In addition to sharing local rewards, what happens if modules also exchange local experience?

In the centralized GAPS algorithm, the parameter updates are computed with the following rule: $\Delta\theta_{oa} = \alpha R(C_{oa} - C_o \pi_\theta(o,a))$, where the counters of experienced observations C_o and chosen actions per observation C_{oa} represent the sum of the agents' individual counters. If there is a way for the agents to obtain these sums through local exchanges, then the assumptions of the theorem are satisfied and we can expect individuals to make the same updates as the global learner in the centralized case, and therefore to converge to a local optimum. Recall that using synchronous, equal and symmetric pairwise disjoint updates guarantees convergence to a common x which is the exact average of initial values of x_i. If at the end of an

episode, each module runs such an agreement algorithm on each of the counters that it maintains, then each counter C^i will converge to $\overline{C} = \frac{1}{N}\sum_{i=1}^{N} C^i$. If this quantity is then substituted into the GAPS update rule, the updates become:

$$\Delta\theta_{oa} = \alpha R\left(\frac{1}{N}\sum_{i=1}^{N} C^i_{oa} - \pi_\theta(o,a)\frac{1}{N}\sum_{i=1}^{N} C^i_o\right) = \frac{1}{N}\alpha R\left(C_{oa} - C_o\pi_\theta(o,a)\right), \tag{6}$$

which is equal to the centralized GAPS updates scaled by a constant $\frac{1}{N}$. Note that synchronicity is also required at the moment of the updates, i.e., all agents must make updates simultaneously, in order to preserve the stationarity of the underlying process. Therefore, robots learning with distributed GAPS using an agreement algorithm with synchronous pairwise disjoint updates to come to a consensus on both the value of the reward, and the average of experience counters, will make stochastic gradient ascent updates, and therefore will converge to a local optimum in policy space.

The algorithm (Alg. 2) requires the scaling of the learning rate by a constant proportional to the number of modules. Since stochastic gradient ascent is in general sensitive to the learning rate, this pitfall cannot be avoided. In practice, however, scaling the learning rate up by an approximate factor loosely dependent (to the order of magnitude) on the number of agents in the system, works just as well.

4 Experiments

We ran the synchronized algorithm described as Algorithm 2 on a case study task of locomotion by self-reconfiguration in 2D, following prior work on GAPS for lattice-based SRMRs [16].

4.1 Lattice-Based Self-Reconfiguring Modular Robots

We use the abstract kinematic model of a 2D lattice-based robot as shown in Fig. 1. The modules are constrained to be connected to each other in order to move with respect to each other; they are unable to move on their own. The robot is positioned on an imaginary 2D grid, viewed from above; and each module can observe at each of its 4 faces (positions 1, 3, 5, and 7 on the grid) whether or not there is another connected module on that neighboring cell. A module (call it M) can also ask those neighbors to confirm whether or not there are other connected modules at the corner positions (2, 4, 6, and 8 on the grid) with respect to M. These eight bits of observation comprise the local configuration that each module perceives.

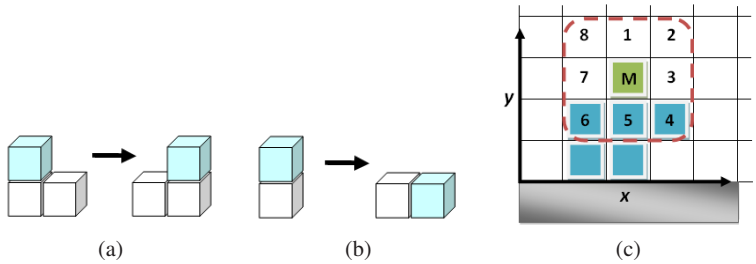


Fig. 1 The sliding cubes model for lattice-based reconfiguration: **(a)** sliding and **(b)** convex transitions. **(c)** The abstract kinematic model of a SRMR on a lattice in 2D.

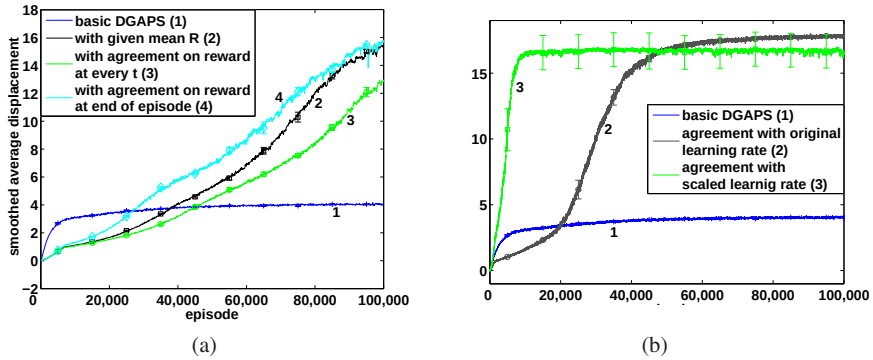
Thus, the module M in the Fig. 1c knows that lattice cells number 4–6 have other modules in them, but lattice cells 1–3 and 7–8 are free space[3]. M has a repertoire of 9 actions, one for moving into each adjacent lattice cell (face or corner), and one for staying in the current position. Modules may not move into lattice positions occupied by other modules or the wall. The robot's abstract kinematics are defined by two possible movements for each module: the sliding and convex transitions shown in Fig. 1a and b. The robot has to remain tethered by at least one module to some point on the base wall.

Each module is limited in its learning ability by the partial observability of the robot configuration, as it only perceives the local window shown. The policy it is learning is represented by a table of parameters, where each entry corresponds to a possible observation-action pair, as described earlier. The learning task is locomotion along the horizontal x axis by self-reconfiguration. Each module i receives a local reward $R^i = x^i_T - x^i_0$ at the end of the episode (time T), which measures its total horizontal displacement.

4.2 Experimental Results

The purpose of the experiments was to discover how agreement on common rewards and/or experience affects the speed and reliability of learning with distributed GAPS. As in earlier distributed experiments in this setting [16], the learning rate α here was initially set at the maximum value of 0.01 and steadily decreased over the first 7,000 episodes to 0.001, and remained at that value thereafter. The temperature parameter β was initially set to a minimum value of 1 and increased steadily over the first 7,000 episodes to 3, and remained at that value thereafter. This encouraged more exploration during the initial episodes of learning.

[3] In Fig. 1c, the module M can only guess that cells 2 and 8 are free space, since no neighbors are available to provide that information. From the point of view of available legal actions, the 'empty' guess will always be correct.



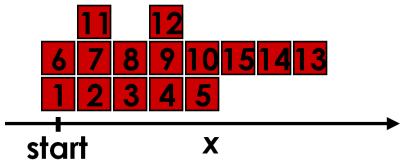
(a) (b)

Fig. 2 Smoothed (100-point window), downsampled average center of mass displacement, with standard error, over 10 runs with 15 modules learning to move right, as compared to distributed GAPS (DGAPS) and DGAPS with given common mean reward (R). **(a)** The modules run the agreement algorithm on R only at each timestep t or to convergence at the end of episodes. **(b)** The modules run the agreement algorithm on both R and experience at end of episodes with original vs. scaled learning rates.

4.3 Agreement on Reward

Figure 2a demonstrates the results of experiments with the learning modules agreeing exclusively on the rewards $\hat{R}$ generated during each episode. The two conditions diverged on the time at which exchanges of reward took place: at the end of the learning episode versus at each timestep t during policy execution. In the latter case, averaging stops with the end of the episode even if agreement has not yet converged to a common value, which results in lower rewards. However, we see that despite this early stopping point, not-quite-agreeing on a common reward is significantly better than using each module's individual estimate, as predicted. Statistical analysis shows no significant difference between DGAPS with given common (mean) R and agreement run at the end of each episode. Agreement (two averaging exchanges at every timestep t) run during locomotion results in a significantly later learning convergence measure for that condition. Basic DGAPS with no agreement fails to learn anything beyond essentially a random policy.

Note DGAPS with agreement at the end of each episode seems to learn better than DGAPS with given common average R for much of the learning curve. There could be a benefit to estimating $\hat{R} \neq R$ based on all-neighbor averaging. Consider 15 modules in a typical non-gait configuration that will nonetheless generate a positive reward and may lead the modules into a bad local optimum in policy space (Fig. 3a). Assume the modules started in a 5x3 rectangular configuration with module 1 at the location labeled 'start'. The average robot displacement here is $R = 2.4$. However, if instead of being given that value, modules actively run the synchronous agreement at the end of this episode, they will eventually arrive at the value of $\hat{R} = 0.63$. This discrepancy is due to the fact that most of the modules, and notably those



mean $R = 2.4$, agreed $\hat{R} = 0.63$

(a)

	no. stuck
DGAPS	10±0.0
DGAPS with common R	0.4±0.2
agreement at each t	0.5±0.2
agreement at end episode	0.2±0.1

(b)

Fig. 3 Effect of active agreement on reward during learning: **(a)** The modules with greatest rewards in such configurations have least neighbors (module 13 has 1 neighbor and a reward of 5) and influence the agreed-upon value $\hat{R}$ the least. The modules with most neighbors (modules 1-10) do not have a chance to get any reward, and influence $\hat{R}$ the most. **(b)** Mean number of stuck configurations, with standard error, out of 10 test runs each of 10 learned policies, after 15 modules learned to locomote eastward for 100,000 episodes.

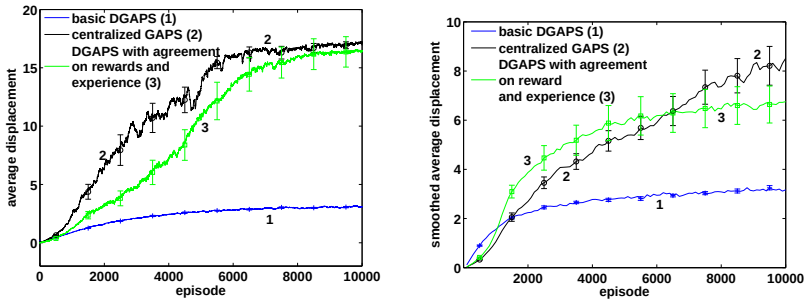
that have on average more neighbors, have received zero individual rewards. Those that have, because they started forming an arm, have less neighbors and therefore less influence on $\hat{R}$, which results in smaller updates for their policy parameters, and ultimately with less learning "pull" exerted by the arm.

Our hypothesis was therefore that, during test runs of policies learned by DGAPS with common reward versus DGAPS with agreement algorithms, we will see significantly more dysfunctional stuck configurations in the first condition. The table in figure 3b details the number of times, out of 10 test trials, that policies learned by different algorithms were stuck in such configurations. Our findings do not support our hypothesis: both types of agreement, as well as the baseline algorithm with common reward, generate similar numbers of non-gait policies.

4.4 Agreement on Reward and Experience

For the set of experiments involving agreement on experience as well as rewards, the learning rate α was scaled by a factor of 10 to approximately account for the averaging of observation and observation-action pair counts[4]. Figure 2b demonstrates that including exchanges of experience counts results in dramatic improvement in how early good policies are found. There is also a significant difference in learning speed between learning with the original learning rate or scaling it to account for averaging of experience. Finally, we see in figure 4a that DGAPS with agreement on both reward and experience learns comparatively just as fast and just as well as the centralized version of GAPS for 15 modules (no significant difference between centralized and agreement groups). However, when we increase the robot size to 20 modules (and episode length to 100 timesteps) in figure 4b, all three al-

[4] This scaling does not in itself account for the difference in experimental results. Scaling up α in the baseline DGAPS algorithm does not help in any way.



(a) (b)

Fig. 4 Comparing centralized GAPS, DGAPS and DGAPS with agreement on experience: **(a)** smoothed (100-point window) average center of mass displacement over 10 trials with 15 modules learning to move right (T=50), **(b)** same for 20 modules (T=100): learning was done starting with $\alpha = 0.2$ and decreasing it uniformly over the first 1,500 episodes to 0.02.

Table 1 Mean number of non-gait local optima, with standard error, out of 10 test trials for 10 learned policies. The starting learning rate was $\alpha = 0.1$ for 15 modules and $\alpha = 0.2$ for 20 modules.

	15 mods, T=50	**20 mods, T=100**
Distributed GAPS (DGAPS)	10±0.0	10±0.0
Centralized GAPS	0.1±0.1	5±1.7
DGAPS with agreement on R^i and C^i	1.1 ±1.0	6±1.6

gorithms yield different reward curves, with agreement-based distributed learning getting significantly less reward than centralized GAPS.

Where does this discrepancy come from? In table 1, we see that learning in a fully distributed way with agreement on reward and experience makes the modules slightly more likely to find non-gait local optima than the centralized version. However, a closer look at the policies generated by centralized learning vs. agreement reveals another difference: gaits found by agreement are more compact than those found by central GAPS. Figure 5a demonstrates in successive frames one such compact gait, as compared to the unfolding gait normally found by centralized GAPS shown in Fig. 5b. In effect, unless updates are made according to Eq. 5, active agreement results in a modified reward function. Whether or not it is preferable to the original will depend on the task at hand.

5 Discussion and Related Work

We have demonstrated that agreement-style exchanges of rewards and experience can be successfully incorporated into gradient-based policy search algorithms to

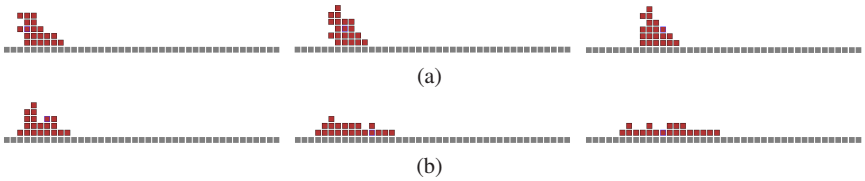


Fig. 5 Screenshots at $t = 5$, 25 and 45, of 20 modules executing **(a)** a compact gait found by learning with agreement on R^i and C^i, and **(b)** an unfolding gait found by centralized GAPS.

enable fully distributed learning with no extra heuristics in lattice-based SRMRs. This result suggests that the centralized algorithms and extensions that were previously presented [15, 16] are applicable to fully distributed systems. With higher numbers of modules and unfavorable initial conditions, there is a significant difference between rewards obtained by centralized learning and distributed learning with agreement, which is primarily due to the kinds of policies that are favored by the different learning algorithms. Non-pairwise agreement on rewards generates an effect where more densely connected learners have more of an influence on the final agreed-upon value. This effectively modifies the reward structure, and therefore, the policy value landscape for local search. However, discussion of reward functions is beyond the scope of this paper.

There is a large body of research in agreement-style algorithms for distributed robotic and sensor network systems (e.g., [8] and citations therein); we cannot do it justice in this space. Our contribution is to combine agreement with a learning algorithms for POMDPs. Our work is very closely related to the distributed optimization algorithm described by Moallemi & Van Roy [10] for networks, where the global objective function is the average of locally measurable signals. They proposed a distributed policy gradient algorithm where local rewards were pairwise averaged, with no assumptions about the policies being learned beyond continuity and continuous differentiability. In contrast, here we are interested in a special case where *ideally all individual policies would be identical*. This limits the class of applicable problems, but learners can take advantage of neighbors' experience as well as rewards. In addition to prior work on distributed GAPS [12], cooperation in distributed RL has been addressed in the fully observable domain through distributed value functions [13] and coordination graphs [5, 7]. By contrast, our approach is applicable to partially observable distributed MDPs.

Practical issues in sharing experience include the required communications bandwidth and message complexity. In most cases, communication is much cheaper and easier than actuation, both in resource (e.g., power) consumption, and in terms of the learning process: learning from others' mistakes and thus reducing potentially dangerous exploration. Additionally, depending on the policy representation, significant downsizing of required communications can be achieved using standard lossless compression schemes, or a more involved distributed protocol such as consensus propagation [11].

6 Conclusion

We have developed a class of algorithms incorporating stochastic gradient ascent in policy space (GAPS) and agreement algorithms, creating a framework for fully distributed implementations of this POMDP learning technique. We demonstrated its success on the learning task of locomotion by self-reconfiguration in 2D simulated SRMRs. Our results suggest that previously published centralized GAPS work may be relevant to fully distributed implementations. In the future, we may wish to develop specific caching and communication protocols for passing just the required amount of information, and therefore requiring less bandwidth, given the policy representation and current experience.

Acknowledgements The authors gratefully acknowledge the support of The Boeing Company.

References

1. J. Baxter and P. L. Bartlett. Infinite-horizon gradient-based policy search. *J. of Artificial Intelligence Res.*, 15:319–350, 2001.
2. D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Athena Scientific, 1997.
3. Y.-H. Chang, T. Ho, and L. P. Kaelbling. All learning is local: Multi-agent learning in global reward games. In *Advances in Neural Information Processing Systems*, volume 16, 2004.
4. F. Fernandez and L. E. Parker. Learning in large cooperative multi-robot domains. *Int. J. of Robotics and Automation*, 16(4):217–226, 2001.
5. C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored MDPs. In *Advances in Neural Information Processing Systems*, volume 14, 2002.
6. J. Hu and M. P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proc. Int. Conf. on Machine Learning*, pages 242–250, 1998.
7. J. R. Kok and N. Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *J. of Machine Learning Res.*, 7:1789–1828, 2006.
8. K. M. Lynch, I. B. Schwartz, P. Yang, and R. Freeman. Decentralized environmental modeling by mobile sensor networks. *IEEE Trans. on Robotics*, 24(3):710–724, 2008.
9. M. J. Matarić. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4(1):73–83, 1997.
10. C. C. Moallemi and B. Van Roy. Distributed optimization in adaptive networks. In *Advances in Neural Information Processing Systems*, volume 15, 2003.
11. C. C. Moallemi and B. Van Roy. Consensus propagation. *IEEE Trans. on Information Theory*, 52(11), 2006.
12. L. Peshkin. *Reinforcement Learning by Policy Search*. PhD thesis, Brown University, 2001.
13. J. Schneider, W.-K. Wong, A. Moore, and M. Riedmiller. Distributed value functions. In *Proc. Int. Conf. on Machine Leanring*, 1999.
14. J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Trans. on Automatic Control*, AC-31(9):803–812, 1986.
15. P. Varshavskaya, L. P. Kaelbling, and D. Rus. Distributed learning for modular robots. In *Proc. Int. Conf. on Robots and Systems*, 2004.
16. P. Varshavskaya, L. P. Kaelbling, and D. Rus. Automated design of adaptive controllers for modular robots using reinforcement learning. *Int. J. of Robotics Res.*, 27(3–4):505–526, 2008.