Ibots Learn Genuine Team Solutions

Cristina Versino and Luca Maria Gambardella

IDSIA, Corso Elvezia 36, CH-6900 Lugano, Switzerland
{cristina,luca}@idsia.ch, http://www.idsia.ch

Abstract. "Ibots" (Integrating roBOTS) is a computer experiment in group learning. It is designed to understand how to use *reinforcement learning* to program automatically a team of robots with a shared mission. Moreover, we are interested in deriving *genuine team solutions*. These are policies whose form strongly depends on the number of robots composing the team, on their individual skills and weaknesses, and on any other mission boundary condition which makes it worth to prefer "at a team level" certain solutions to others. The Ibots learn to accomplish the integration mission by means of a reinforcement signal which measures their performance as a team. This form of payoff leads to genuine team solutions. Benefits and drawbacks of using a single team payoff as opposed to individual robot payoffs are discussed.

1 Introduction

Reinforcement learning [1] provides us with a framework to achieve self-programming, adaptive robots. In *single robot* missions, the reinforcement signal directly evaluates the behavior of the only robot in charge of the task. The picture changes as *many robots* are acting synchronously, with no knowledge about teammate activities. In this scenario, if the reinforcement signal reflects the whole team performance, each single robot is faced with the problem of deciding to what extent its own behavior has contributed to the overall team's good or bad score: this is the *robot credit assignment problem*.

The robot credit assignment problem can be bypassed in at least two ways.

A *first* way is to enable *communication* between teammates [2, 3, 4]. If a robot is *aware* of other robots' perceptions and actions, then it is in a position to make sense out of a global team payoff. But communication is not always possible technically and it tends to become a bottleneck as the team size increases [5].

A *second* way of avoiding the robot credit assignment problem is to measure *each robot's individual performance* instead of team performance. In [6, 7] this idea is applied to the training of a group of real robots in a puck collection task. Each robot in the team learns a personal policy through individual payoff. For example, a robot is rewarded whenever it either grasps a puck or drops a puck at a home area. In this framework, a single robot is not interested in the performance of its teammates, because it addresses the mission in an individualistic sense. We see *two* drawbacks in this approach. *First:* its assumption is that team performance indirectly increases because individual performance increases. However, if the robots do not learn the task at a *similar pace*, it cannot

be guaranteed that each robot will learn and participate to the mission. If not all robots learn how to contribute to the mission, the team performance will be suboptimal. As an example, suppose that in the puck collection task one robot in the team manages to learn the individually optimal policy after a few trials. This "super-robot" will collect most of the pucks by itself, diminishing the learning opportunities of its teammates because pucks are a *limited, shared resource*. *Second:* where do the policies learnt by the robots converge? They will converge to the optimal policy for a robot carrying out the mission by itself. The robots will behave as "clones" of *a robot designed to work alone*. We feel this violates the spirit of team learning, which should be aimed instead at producing *genuine team solutions*.

Genuine team solutions are policies whose shape is strongly influenced by the number of robots composing the team, by each robot's skills and weaknesses, and by any other mission boundary condition which is relevant for discriminating "at a team level" some good solutions against some others.

To obtain truly team solutions, one should use team payoffs at the price of dealing with the ambiguity posed by the robot credit assignment problem.

Experiments along this line are illustrated in recent works [8, 9, 10]. In [8] a team of simulated agents learns signaling behaviors to efficiently solve an object-gathering task in an unknown and changing environment. The reinforcement signal is based on the total time needed by the team to gather all the objects in the workspace. Experiments are carried out under several conditions: with teams of different size, with a variable number of objects, and with different object distributions. Statistical analysis of the results shows that the team is able to discover a near-to-optimal signaling policy given specific mission conditions. In [9] a team of Q-learning agents is engaged in the real-world problem of elevator dispatching. Each agent is responsible for controlling one elevator car. Two different control architectures are tested. In the *parallel* architecture, the agents share a single neural network which models a common policy: this allows the agents to learn from each others experiences but forces them to use identical policies. In the *decentralized* architecture, the agents learn personal networks, which allow them to specialize their control policies. In both architectures, the team receives as global payoff the sum of squared wait times of passengers. Results obtained in simulation surpass the best known heuristic elevator control algorithm.

This paper presents "Ibots" (Integrating roBOTS) [10], a computer experiment in collective robotics designed to understand how to use reinforcement learning to program automatically a team of robots with a shared mission. The "integration" mission is an artificial task for robots, but it addresses several issues commonly arising in group learning, such as: *sharing a limited resource* in a way which is beneficial for the team; learning *public* or *private* policies; learning by trial-and-error from a *global team payoff*. Experimental results show that the Ibots learn to adapt their behavior to the actual team size and to different mission boundary conditions, as well as to each robots's special skill or limitation. The robot credit assignment problem (which arises when the Ibots learn private policies *without communication*) is handled with rules analogous to those used

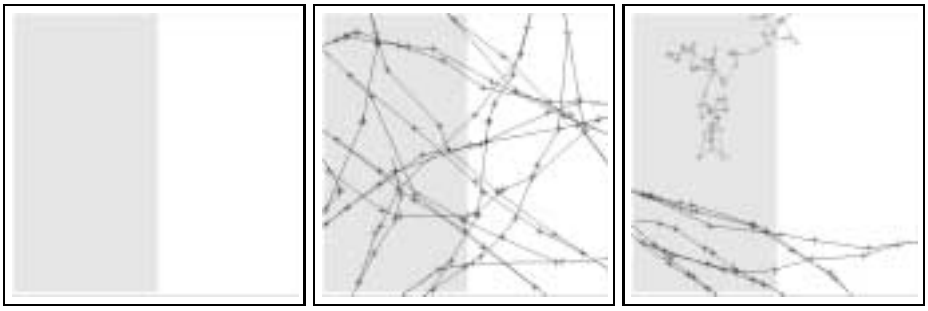

Fig. 1. (Left) The squared arena and the "half-full" *Region*: $I = 0.49$. The arena side is 500 unit long. (Middle) Trajectory of one Ibot running a solitary trial with $N_{max} = 100$ and $Prog = (30°, 200)$. Crosses indicate sampled points. The result is $N_{in} = 52$, $\hat{I} = 0.52$. (Right) Trajectories of two Ibots running a shared trial with $N_{max} = 100$, starting from a scattered configuration and running private control programs: $Prog^1 = (180°, 50)$ (gray trace), $Prog^2 = (20°, 150)$ (black trace). The result is $N^1_{in} = 55$, $N^2_{in} = 30$, $\hat{I} = 0.85$. Ibot 1 took $N^1_{sam} = 61$ samples, Ibot 2 took $N^2_{sam} = 39$.

in connectionist reinforcement learning to solve the network structural credit assignment problem [11]. After all, a neural net is a good example of a set of partially independent agents which learn to act well as a team. In a similar way, the Ibots manage to learn genuine team solutions.

2 Ibots

The *mission* for the Ibots is to guess the integral I ($0 \leq I \leq 1$) of an arbitrary gray *Region* drawn on the white ground of a squared arena (Fig. 1, left).

How are robots turned into Ibots? Let us first consider the case of a team composed of one Ibot.

2.1 One Ibot

The Ibot's *control program* $Prog$ lets it explore the arena while sampling the ground color. By activating $Prog$, the Ibot performs a *trial* run (Fig. 1, middle).

A trial starts from a random location in the arena. It is a sequence of N_{max} *elementary movements* separated by stops. Whenever the Ibot stops, it samples the ground color. A "gray" reading gives evidence for the sample to be "inside *Region*", a "white" reading is interpreted as "outside *Region*". At the end of the trial, the Ibot returns the number N_{in} of samples inside *Region*. $\hat{I} = N_{in}/N_{max}$ is its estimate of I at this trial. $E = |I - \hat{I}|$ is the error in the estimate.

How are elementary movements generated? The Ibot control program $Prog$ depends stochastically on two parameters $(\alpha_{prog}, \delta_{prog})$ which remain fixed during a trial. α_{prog} is used to generate a rotation instruction for the Ibot, while δ_{prog} induces a translation. The semantics of α_{prog} and δ_{prog} is as follows. First, a number α is drawn from a uniform distribution in $[-\alpha_{prog}, +\alpha_{prog}]$. This is

interpreted by the Ibot as: "Rotate α degrees.". Second, a number δ is drawn from a uniform distribution in $[0, \delta_{prog}]$. The interpretation for δ is: "Translate δ units in your current heading direction, calling the *bumping rule* if necessary.". The bumping rule is called when the Ibot meets the arena border before having covered the whole distance δ. In this case, the Ibot rotates 180° and covers the remaining distance. The bumping rule can be called recursively.

In our computer experiment, a rotation instruction is executed by the Ibot in one time unit whatever the rotation angle α; the execution time of a translation instruction is directly proportional to the distance δ.

Finally, both program parameters α_{prog} and δ_{prog} take values in a finite range: $0 \leq \alpha_{prog} \leq \alpha_{max}$ and $0 \leq \delta_{prog} \leq \delta_{max}$.

2.2 A Team of Ibots

Each Ibot i ($i = 1, \ldots, N_{ibots}$) being equipped with a control program $Prog^i = (\alpha^i_{prog}, \delta^i_{prog})$, there are several ways of generalizing from the single Ibot case to the team case (Fig. 1, right). We have considered both the cases where the $Prog^i$s may be *public* or *private*. Public means that all Ibots share the same $Prog$ (i.e., all $Prog^i$ are equal), while private means that each Ibot works with a personal, different $Prog^i$.

The Ibots activate the $Prog^i$s in parallel to run a team trial. At the beginning of the trial, the Ibot locations are chosen at random in the arena, and these are either *clustered* or *scattered*. In the clustered configuration, all Ibots have the same initial position and orientation; in the scattered configuration, they have different positions and orientations. During a trial, the Ibots are granted a total of N_{max} elementary movements to collect N_{max} samples of the ground color *overall*: *samples are the team limited and shared resource*. Whenever an Ibot stops to take a sample, it is allowed to do so only if less than N_{max} samples have been taken by the team so far. Notice that when the Ibots work with private $Prog^i$s, each Ibot i collects a different number of samples N^i_{sam}: Ibots with small δ^i_{prog}s traslate for short distances and take, on the average, more samples than Ibots with larger δ^i_{prog}s. At the end of the trial, each Ibot i returns the number N^i_{in} of samples it counted inside *Region*. These contributions are summed in $N_{in} = \sum_i N^i_{in}$, leading to the team integral estimate $\hat{I} = N_{in}/N_{max}$, the error being $E = |I - \hat{I}|$.

Finally, Ibots are *immaterial*, they do not collide when their trajectories intersect.

3 Programmed Ibots vs. Learning Ibots

The goal of group learning is to find control programs $Prog^i$s which lead to estimates $\hat{I}$ close to I. As each program depends on its parameters α^i_{prog} and δ^i_{prog}, *the target of learning is to discover "good" pairs* $(\alpha^i_{prog}, \delta^i_{prog})$.

Notice that we know a *general* solution, namely:

$$\forall i : \begin{cases} \alpha^i_{prog} = 180^\circ \\ \delta^i_{prog} = \text{“length of arena diagonal”} \end{cases} \quad (1)$$

With this choice of parameters, the Ibots perform a pseudo-random motion and take samples *uniformly distributed* in the arena. This brings us to the hypothesis of the Monte Carlo method [12] for integration. This states that, by drawing N_{max} points from a uniform distribution in the arena, the error E in the estimate of the integral is probabilistically bounded by N_{max}:

$$P\left\{E \leq \frac{1}{\sqrt{N_{max}}}\right\} \geq 0.9999 \quad (2)$$

For example, by drawing 100 points, the error in the estimate will not exceed 0.1. Given N_{max}, this result quantifies the *admissible error* for the Ibots mission. We call the control programs defined by Eq. 1 the “programmer solution”, because it reflects, in our opinion, the way a programmer would address this robot programming task: by looking for a *general* solution, which will work whatever the number of Ibots in the team, independently of their starting configuration in the arena. Though appealing, we are not interested in this *a priori* solution. Rather, we are looking for *real team solutions* established through experience. These should depend on the number of Ibots, on their initial configuration, and on their specific skills when these latter are no longer homogeneous.

4 Learning to Be Good Ibots

“Good” $Prog^i$s are programs which lead to *admissible* and *stable* estimates of I. The Ibots learn good $Prog^i$s by reinforcement through a sequence of trials. Each time instant t corresponds to a team trial. Let $Prog^i(t) = (\alpha^i_{prog}(t), \delta^i_{prog}(t))$ be Ibot i control program at time t. The following 4 steps are repeated forever.

1. *Each Ibot “i” independently generates a new, tentative control program* $New^i(t) = (\alpha^i_{new}(t), \delta^i_{new}(t))$ *by slightly modifying* $Prog^i(t)$.

   Given:

   $$\alpha^i_{temp}(t) = \alpha^i_{prog}(t) + \alpha^i_{rand}(t) \cdot \alpha_{step}$$
   $$\delta^i_{temp}(t) = \delta^i_{prog}(t) + \delta^i_{rand}(t) \cdot \delta_{step}$$

   it is:

   $$\alpha^i_{new}(t) = \begin{cases} 0^\circ & \text{if } \alpha^i_{temp}(t) < 0^\circ \\ \alpha_{max} & \text{if } \alpha^i_{temp}(t) > \alpha_{max} \\ \alpha^i_{temp}(t) & \text{otherwise} \end{cases}$$

$$\delta^i_{new}(t) = \begin{cases} 0 & \text{if } \delta^i_{temp}(t) < 0 \\ \delta_{max} & \text{if } \delta^i_{temp}(t) > \delta_{max} \\ \delta^i_{temp}(t) & \text{otherwise} \end{cases}$$

where:

- $\alpha^i_{rand}(t)$ and $\delta^i_{rand}(t)$ are uniform random numbers in $[-\rho(t), +\rho(t)]$ (see below for the definition of $\rho(t)$);
- α_{step} and δ_{step} are constants.

2. *The Ibots collectively carry out a trial with the newly generated programs $New^i(t)$s.*

   At the beginning of the trial, the Ibots are positioned at a random configuration. At the end of the trial, each Ibot returns $N^i_{in}(t)$. The team integral estimate is $\hat{I}(t) = N_{in}(t)/N_{max}$, with $N_{in} = \sum_i N^i_{in}$. The error in the estimate $E(t)$ is:

$$E(t) = \frac{|I - \hat{I}(t)|}{\max(I, 1 - I)}$$

3. *The team reinforcement signal $R(t)$ is computed and communicated to each Ibot.*

   $R(t)$ is defined as the difference between the team errors in two successive trials:

$$R(t) = E(t-1) - E(t) \tag{3}$$

   $E(t-1)$ can be thought of as a naïve predictor of $E(t)$.

4. *Each Ibot "i" independently computes a modification for its control program and updates it.*

   The modification is:

$$\Delta\alpha^i_{prog}(t) = \epsilon \cdot R(t) \cdot (\alpha^i_{new}(t) - \alpha^i_{new}(t-1)) \tag{4}$$
$$\Delta\delta^i_{prog}(t) = \epsilon \cdot R(t) \cdot (\delta^i_{new}(t) - \delta^i_{new}(t-1)) \tag{5}$$

   where ϵ, a real parameter, is the learning rate.
   Given:

$$\alpha^i_{temp}(t+1) = \alpha^i_{prog}(t) + \Delta\alpha^i_{prog}(t)$$
$$\delta^i_{temp}(t+1) = \delta^i_{prog}(t) + \Delta\delta^i_{prog}(t)$$

the updated control programs is:

$$\alpha^i_{prog}(t+1) = \begin{cases} 0^\circ & \text{if } \alpha^i_{temp}(t+1) < 0^\circ \\ \alpha_{max} & \text{if } \alpha^i_{temp}(t+1) > \alpha_{max} \\ \alpha^i_{temp}(t+1) & \text{otherwise} \end{cases}$$

$$\delta^i_{prog}(t+1) = \begin{cases} 0 & \text{if } \delta^i_{temp}(t+1) < 0 \\ \delta_{max} & \text{if } \delta^i_{temp}(t+1) > \delta_{max} \\ \delta^i_{temp}(t+1) & \text{otherwise} \end{cases}$$

The description of the algorithm is completed by the following remarks and definitions.

- When the Ibots work with public control programs, only one Ibot generates the tentative program $New^i(t)$ at step 1; then, $New^i(t)$ is communicated to the other team members.
- About the error measure $E(t)$: by dividing $|I - \hat{I}(t)|$ by $\max(I, 1-I)$, $E(t)$ varies between 0 and 1, no matter what the value of I. As a consequence, the reinforcement signal also varies in a fixed interval, namely $[-1, +1]$.
- $E(-1) = E(0)$. At trial 0, we assume that the expected error $E(-1)$ is equal to the measured error $E(0)$. This implies $R(0) = 0$, so no change is made to $Prog(0)$.
- $\rho(t) = max(|R(t-1)|, \rho_c)$. The amount of variation in the new control programs is proportional to the absolute value of the reinforcement signal at previous trial. This is to enhance the tendency of escaping from programs with unpredictable performance, and, viceversa, to favor the convergence towards programs with stable performance. ρ_c is a positive constant which maintains a minimal level of exploration in program space when $R(t-1) = 0$.

In the reinforcement learning panorama, this set-up corresponds to a *nonassociative, immediate* reward learning problem running on a distributed system. The closest analogy is with a neural net [11] with no inputs from the environment other than the performance signal itself, and using an adaptive critic to predict its performance. The learning algorithm is expected to guide the team towards admissible and stable control programs.

5 Experiments

On the "half-full" *Region* ($I = 0.49$) of Fig. 1, we have run repeated learning experiments with Ibots' teams of increasing size ($N_{ibots} = 1, \ldots, 14$), with public or private $Prog^i$s, and starting from clustered or scattered configurations. We have also considered Ibots with heterogeneous skills due to differences in sensing and acting capabilities. In all experiments we have set: $N_{max} = 100$ (the corresponding admissible error being 0.1), $\alpha_{max} = 180^\circ$ and $\delta_{max} = 500$ (length of arena side), $\alpha_{step} = \alpha_{max}/10$ and $\delta_{step} = \delta_{max}/10$, and, for each Ibot i, $Prog^i(0) = (\alpha^i_{prog}(0), \delta^i_{prog}(0)) = (0^\circ, 0)$.

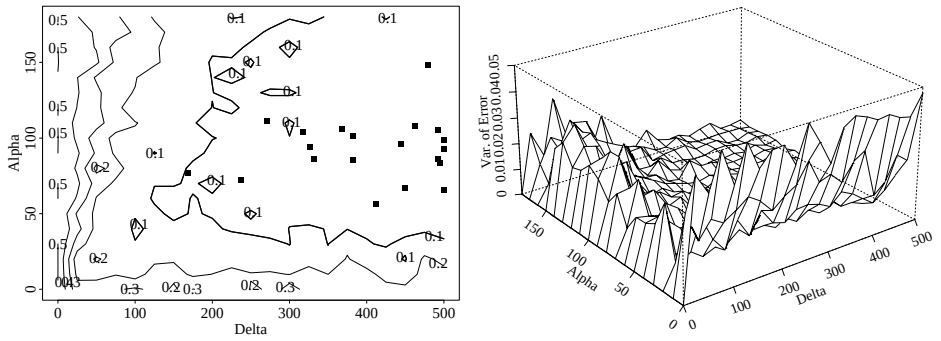

Fig. 2. One Ibot. (Left) $\mu_{prog}(E)$ and convergence points for 20 learning experiments. (Right) $\sigma^2_{prog}(E)$.

To examine the form of learnt programs, a learning experiment was stopped either after having obtained 10 consecutive admissible estimates, or after a predefined number of trials, depending on which of these two events occured first. Programs learnt by applying this stopping criterion are called the *convergence points* of the learning experiment.

5.1 One Ibot

The single Ibot experiment is a point of reference for comparing results obtained with teams of Ibots. It requires to discover a pair $(\alpha_{prog}, \delta_{prog})$ which produces admissible and stable integral estimates. As in this case the program space is bidimensional, one can explore it in a systematic way to test the quality of a significant number of programs. Thus, before starting the learning experiments, we have run background trials with different combinations of α_{prog} and δ_{prog} values. Each combination program was tested on N_{trials} different trials to have a sample of integral estimates $\hat{I}_k$. Then, for each program we computed the *mean of errors* $\mu_{prog}(E)$ and the *variance of errors* $\sigma^2_{prog}(E)$:

$$\mu_{prog}(E) = \frac{1}{N_{trials}} \sum_k |I - \hat{I}_k| = \frac{1}{N_{trials}} \sum_k E_k$$

$$\sigma^2_{prog}(E) = \frac{1}{N_{trials}} \sum_k (E_k - \mu_{prog}(E))^2$$

$\mu_{prog}(E)$ measures the accuracy of the estimates, while $\sigma^2_{prog}(E)$ measures their stability.

Figure 2 shows plots of $\mu_{prog}(E)$ (left) and $\sigma^2_{prog}(E)$ (right) in the Ibot program space. On the $\mu_{prog}(E)$ plot we have highlighted the contour lines of level 0.1: these lines identify the space of admissible programs. Notice that these programs are also stable. Most of them are distant from the "programmer solution"

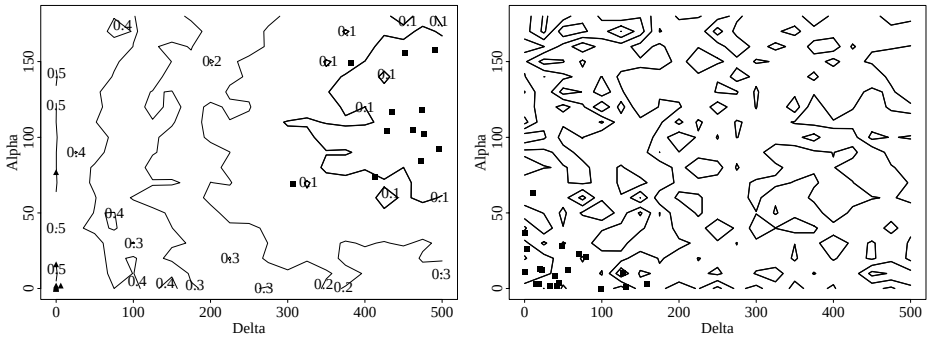

Fig. 3. A team of 14 Ibots with public control programs. $\mu_{prog}(E)$ and convergence points for 20 learning experiments for clustered configurations (left) and for scattered configurations (right).

$(180°, 700)$. From the $\sigma^2_{prog}(E)$ plot we also remark that not only admissible programs are stable. For example, all "staying in place" programs ($\delta_{prog} = 0$) have a very predictable performance. This makes the learning task more difficult.

The convergence points of 20 learning experiments have been indicated as squares on the left plot of Fig. 2. Learning stopped in all cases before the limit of 1000 trials. All experiments ended inside the space of admissible and stable programs.

5.2 Teams of Ibots

The form of admissible and stable programs completely changes for teams of Ibots.

Public Control Programs. The first case study we have addressed is that of Ibots equipped with public control programs. As in the single Ibot case, the program space is bidimensional, so we first ran background trials (with no learning) for teams of increasing size, both for the clustered and the scattered configuration. The results for the largest team of 14 Ibots are shown in Fig. 3. The left plot is the contour plot of $\mu_{prog}(E)$ for the clustered configuration, while the right plot shows $\mu_{prog}(E)$ for the scattered configuration. On the left plot, the bold line delimits the space of admissible programs; for the scattered configuration, all programs are admissible.

By comparing these results with those of Fig. 2 (left), one observes how the single Ibot's solution space "shrinks" or "expands" depending on whether the Ibots are started in the clustered or in the scattered way. Why?

First, consider the clustered configuration starting condition. As the Ibots begin a trial from the same position and with the same orientation, they have to disperse in the arena in order to explore it. Moreover, the number of samples they are allowed to take as individuals decreases as the team size increases,

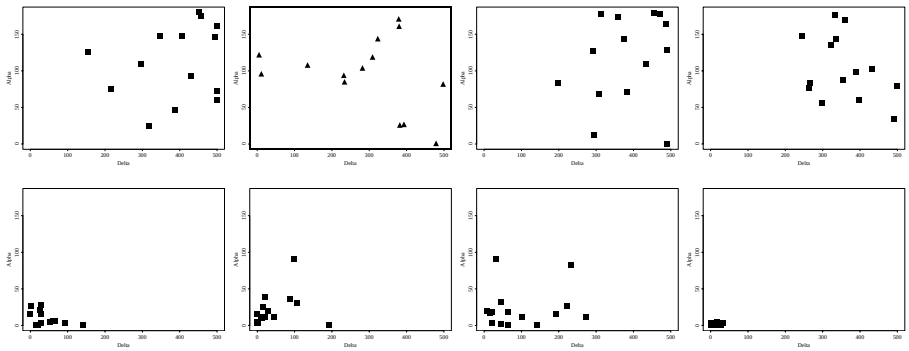

Fig. 4. A team of 14 Ibots with private control programs. Convergence points for 4 learning experiments for clustered configurations (first row) and for scattered configurations (second row). In each plot, the vertical axis is indexed by α_{prog}, the horizontal axis by δ_{prog}.

because the samples budget N_{max} remains the same whatever the team size. As a consequence, in a large team, each Ibot is granted fewer samples and fewer elementary movements to disperse in the arena. Given this constraint, the only way of achieving dispersion in few movements is through control programs with large variability both in translation and in rotation. *In conclusion, most of the solutions which are valid for the single Ibot would not work for this team.*

Second, consider the opposite case where the Ibots start a trial from scattered positions. As they are already uniformly distributed in the arena, any kind of motion program would lead to admissible estimates. *Most of this team solutions would not be admissible for the single Ibot.*

Figure 3 also reports the convergence points of 20 learning experiments for both initial configuration types. Programs which did not converge to admissible and stable estimates within the time limit of 2000 trials are represented by triangles. Not surprisingly, all the experiments for the scattered configuration converged very rapidly (right). On the contrary, for the clustered configuration, not all experiments managed to converge to admissible programs within the predefined time limit (left). This is due to the fact that the Ibots initial program is $Prog^i(0) = (0°, 0)$, a bad but very stable program.

Private Control Programs. Figure 4 shows a representative set of learning experiments performed with a team of 14 Ibots working with private control programs and started from clustered (first row) or scattered configurations (second row). Each plot shows the programs learnt by the team within the time limit of 20000 trials. Essentialy, these solutions are similar to those obtained with public control programs: clustered Ibots need large variability in angle and translation, while scattered Ibots don't. This uniformity is not surprising, because the integration task does not require differentiation in behavior as long as the Ibots have homogeneous skills.

From the point of view of learning, the main difference between dealing with

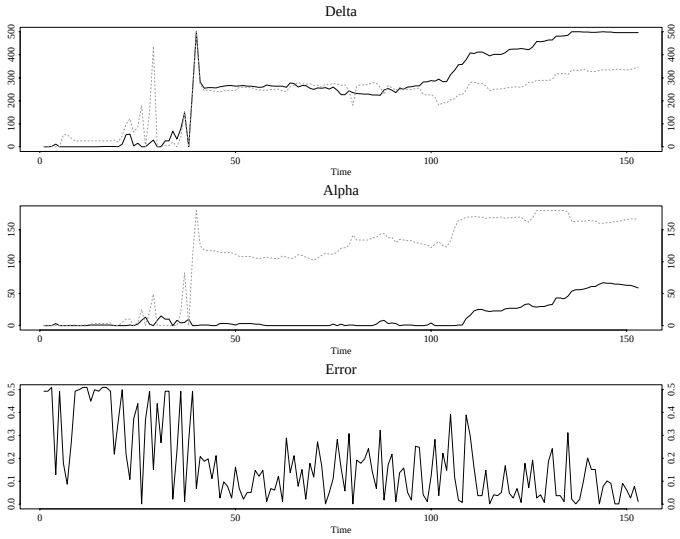

Fig. 5. Robot credit assignment problem for a team of two Ibots: Ibot 1 (dashed line) and Ibot 2 (solid line). Time evolution of δ^i_{prog}s (first plot), of α^i_{prog}s (second plot), and of the error E (third plot).

a single public program or with many private programs is the robot credit assignment problem. Figure 5 illustrates the robot credit assignment problem for a team of two Ibots. The first and the second time plot present the evolution of the Ibots' δ^i_{prog}s and α^i_{prog}s parameters, respectively; the third time plot shows the error E in the integral estimate. Around time 30, Ibot 1 (dashed line) has already acquired an admissible program ($Prog^1 = (49°, 437)$), but this does not appear at level of team performance because Ibot 2 (solid line) is still locked to the initial "staying in place" program. Consider also that Ibot 2's contribution to the team integral weighs more than Ibot 1's contribution, because Ibot 2 takes more samples. This is also the cause for the non-converging experiment of Fig. 4 (first row, second plot), where a minority of Ibots translating for short distances damage the team performance. To improve this state of affairs, Ibot 1 (Fig. 5), first backtracks from its admissible program, then relearns at a similar pace with Ibot 2.

Private Control Programs for Heterogeneous Ibots. As a last experiment, we forced the learnt control programs to specialization by differentiating the Ibots' skills. Figure 6 refers to a learning experiment with a team of two heterogeneous Ibots running private control programs. Ibot 1 (dashed line) translates four times as fast as Ibot 2 (solid line). Moreover, Ibot 2 is *blind*: its ground color sensor reads "white" whatever its position in the arena.

The strategy discovered by the team to provide admissible and stable estimates is clear from the δ^i_{prog}s and α^i_{prog}s plots of Fig. 6. The blind Ibot minimizes its catastrophic contribution to the team integral estimates by travelling long

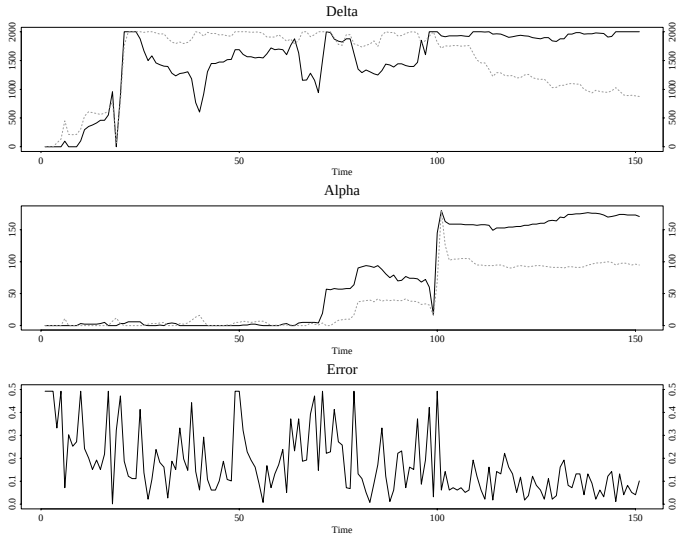

Fig. 6. A team of two heterogeneous Ibots: Ibot 1 (dashed line) and Ibot 2 (solid line). Time evolution of δ^i_{prog}s (first plot), of α^i_{prog}s (second plot), and of the error E (third plot).

δ^1_{prog}	α^1_{prog}	δ^2_{prog}	α^2_{prog}	$\mu(N^1_{sam})$	$\sigma(N^1_{sam})$
565	95	1528	179	85.5	1.5
876	95	2000	171	85.9	1.7
361	176	1549	2	88.9	1.3
537	165	1646	102	86.9	2.0
422	75	1061	66	84.1	0.5

Table 1. (Columns 1–4) Programs learnt by the heterogeneous team in 5 repeated experiments (one experiment per row). (Columns 5 and 6) Mean and standard deviation of the number of samples taken by Ibot 1 over 10 trials.

distances ($\delta^2_{prog} = 2000$, having set $\delta_{max} = 2000$ for this particular experiment). Observe that the error E stabilizes to low values only when the difference between δ^1_{prog} and δ^2_{prog} is sufficiently large. Still, Ibot 1 can afford a parameter of $\delta^2_{prog} = 800$ because it moves very fast.

Table 1 reports more programs learnt by this team in 5 repeated experiments. Ibot 1 always travels for shorter distances than Ibot 2. In all experiments, the balance between δ^1_{prog} and δ^2_{prog} is such that Ibot 1 consistently manages to collect at least 85 out of the 100 samples available to the team.

6 Conclusions

The aim of the Ibots experiment was to understand how to use reinforcement learning to program automatically a team of robots with a common mission. In addition, we wanted to derive *genuine team solutions*.

The learning scenario for the Ibots is applicable to other missions because it relies on weak assumptions. A *team reinforcement signal* evaluates the behavior of the group as a whole. A *limited, common resource* constrains the Ibots, but there is no a priori rule to decide how this resource should be shared. When working with private control programs, the Ibots are *unaware* of teammate control programs.

As a general conclusion, experiments have demonstrated that different mission conditions require different control programs, and that a simple reinforcement learning procedure can find the solutions. The key issue is to optimize team performance instead of individual performance.

As far as the specific Ibots experiment is concerned, we cannot claim that the solutions discovered by the learning procedure were completely unexpected. However, as robot programmers, we have only a limited intuition on the form of the solutions which are more appropriate for specific mission conditions. This motivates the use of learning as an alternative to handcrafting the programs for the team. *Second*: in general, a program which is admissible for a single Ibot is not admissible for a team of Ibots, and viceversa. Thus, we cannot simply find a solution for one Ibot and clone it n times, n being the team size. The form of the solution to a problem changes as the number of "problem solvers" changes. Moreover, the robots become aware of this fact only if they are confronted with their performance as a team. On the contrary, a group of robots learning from individual payoffs would ignore opportunities which become evident only if the task is considered at a team level. *Third*, the space of admissible programs strongly depends on the number of Ibots involved in the mission and on their initial configuration in the arena. The admissibility space "shrinks" when the team size grows and the Ibots are started in a clustered configuration. On the contrary, the admissibility space "enlarges" when the Ibots are started in a scattered configuration. The mission becomes easier in this case because, by initially distributing the Ibots at random in the arena, we bring them close to the problem solution; a team of individualistic robots would not be aware of this opportunity. *Fourth*, when the Ibots work with private control programs, the robot credit assignment problem arises, resulting in longer learning time. Interestingly, the robot credit assignment problem forces the Ibots to learn admissible programs at a similar pace, to prevent "slow" learners from jeopardizing the team mission. *Fifth*, the robot credit assignment problem vanishes when the Ibots learn a public policy, and the learnt policy is still a genuine team solution. The possibility of learning a single public program instead of several private programs should be not overlooked in missions where specialization of robot behavior is not required: the time necessary for the team to learn a public program is much shorter. Finally, *sixth*, the Ibots with their heterogeneous acting and sensing capabilities manage to specialize private control programs. They take advantage of individual skills

and minimize the impact of individual weaknesses. In this way, the "blind" Ibot and the "fast" Ibot find a solution which is good for the team. This is possible because Ibots do not care about individual performance.

Acknowledgements

Cristina Versino is supported by the project No. 2129-042413.94/1 of the Fonds National de la Recherche Scientifique, Berne, Suisse. We thank an anonymous reviewer for helpful comments.

References

1. A.G. Barto, R.S. Sutton, and C.J.C.H. Watkins. Learning and sequential decision making. Technical Report COINS-89-95, Dept. of Computer and Information Science, University of Massachusets, Amherst, 1989.
2. L.E. Parker. The effect of action recognition and robot awareness in cooperative robotic teams. In *IROS95, IEEE/RSJ International Conference on Intelligent Robots and Systems, Pittsburgh, PA, August*, volume 1, pages 212–219, 1995.
3. L.E. Parker. ALLIANCE: An architecture for fault tolerant, cooperative control of heterogeneous mobile robots. In *IROS94, IEEE/RSJ International Conference on Intelligent Robots and Systems, Munich, Germany, September*, pages 776–783, 1994.
4. M.J. Matarić, M. Nilsson, and K.T. Simsarian. Cooperative multi-robot box-pushing. In *IROS95, IEEE/RSJ International Conference on Intelligent Robots and Systems, Pittsburgh, PA, August*, 1995.
5. C.R. Kube and H. Zhang. Collective robotics: From social insects to robots. *Adaptive Behavior*, 2(2):189–218, 1994.
6. M.J. Matarić. Interaction and intelligent behavior. Technical Report AI-TR-1495, MIT Artificial Intelligence Lab, Boston, 1994.
7. M.J. Matarić. Learning in multi-robot systems. In G. Weiss and S. Sen, editors, *Adaptation and Learning in Multi-Agent Systems*, volume 1042, pages 152–163. Springer-Verlag, Lecture Notes in Artificial Intelligence, 1996.
8. A. Murciano and J. del R. Millán. Learning signaling behaviors and specialization in cooperative agents. *Adaptive Behavior*, 5(1):5–28, 1997.
9. R.H. Crites and A.G. Barto. Improving elevator performance using reinforcement learning. In D.S. Touretzky, M.C. Mozer, and M.E. Hasselmo, editors, *Advances in Neural Information Processing Systems 8*, pages 1017–1023, Cambridge MA, 1996. MIT Press.
10. C. Versino and L.M. Gambardella. Learning real team solutions. In G. Weiss, editor, *DAI Meets Machine Learning*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 1997. In press.
11. A.G. Barto, R.S. Sutton, and C.W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, 13:835–846, 1983.
12. J.M. Hammersley and D.C. Handscomb. *Monte Carlo Methods.* Methnen & Co., London, 1964.