Teaching with RoboCup

Jacky Baltes[1], **Elizabeth Sklar**[2] **and John Anderson**[1]
[1]Department of Computer Science
University of Manitoba
Winnipeg, Manitoba, R3T 2N2, Canada
jacky,andersj@cs.umanitoba.ca
[2]Department of Computer Science
Columbia University
New York, NY, 10027, USA
sklar@cs.columbia.edu

Abstract

This paper describes the design and implementation of a new, simplified, entry-level RoboCup league and its integration into an introductory robotics and artificial intelligence curriculum. This *E-League* allows teams to focus on individual aspects such as hardware platform development or multi agent coordination, because the league provides modular solutions for several components and lets teams concentrate on chosen area(s) instead of requiring that all teams solve all aspects of a coordinated RoboCup team.

Introduction

This paper describes the design and implementation of a new, simplified, entry-level RoboCup league (Anderson *et al.* 2003) and its integration into an introductory robotics and artificial intelligence curriculum. RoboCup, initiated in 1997, was designed to bring together robotics and artificial intelligence researchers world-wide by providing a common problem that would require advances in many fields and a collective approach to solve (Kitano *et al.* 1997). The chosen arena was robotic soccer, currently played by autonomous robots in several categories — *leagues* — which vary in physical size, cost, type of hardware platform and approaches to vision and software control. In 2000, the RoboCupJunior division was formed, with the goal of introducing young students (primary through high school) to RoboCup and providing them with an exciting and motivating way to learn about technology through hands-on experiences (Sklar, Eguchi, & Johnson 2002).

Four years later, there is now a growing population of RoboCupJunior "graduates" who are interested in continuing with their efforts but do not have a place within the initiative because they do not have the resources required to enter the senior leagues. Some of these students may be attending a university that has an existing RoboCup team, so that by the time they are advanced undergraduates, they will perhaps have an opportunity to participate on a senior league team. But other students attend universities where there is no RoboCup senior team and/or no robotics lab capable of producing one. For these students, participation as undergraduates is not an option. We were thus motivated to create an entry-level league for RoboCup, not only RoboCupJunior veterans but also for students new to RoboCup.

In developing this new league, which we call the *E-League*, we have discovered that in addition to the practical rationale of providing a stepping stone for undergraduate students, the new setup also offers unique research challenges and teaching opportunities. Since the league provides a common, modular solution to each of the team aspects (e.g., vision, communication and control), teams can choose one component as their focus and use the standard solution for the other components. This means that teams could choose an off-the-shelf hardware platform, such as LEGO Mindstorms, and put their efforts into designing artificial intelligence techniques such as path-planning or multi agent techniques such as coordination.

This paper first outlines our efforts so far in developing the league, detailing the setup and outlining the rules. The second part of the paper discusses use of the league within an introductory course in embodied agents.

Technical description

The initial goal of the league described herein is to provide a stepping stone from RoboCupJunior (RCJ) to participation in the senior Small-Size League (SSL) or Mid-Size League. There is significant leap in both expertise and resources necessary to be a competitive entrant in these RoboCup leagues as compared to RCJ. We estimate that it takes approximately two years to build a RoboCup team from scratch, which is a large time commitment for undergraduate students.

Another problem is the sophistication of the robots used in the SSL and their cost. A typical SSL team has robots that have omni-directional drives, dribble bars and powerful kicking mechanisms. Such a robot has four to six high quality motors and a powerful on-board processor to control them. Each one of these robots costs around US$3,000. Added to this is the cost of high quality video cameras. These and other expenses typically drive the cost for a team to around US$20,000–US$30,000.

For these reasons, it is difficult for students that are too old for RCJ but are not part of an already established senior team to either enter or continue involvement with RoboCup. The E-League is intended to provide a scaled-down and less expensive version of the SSL for entry-level students. Competition in this new league will provide students with a problem-solving experience that will also be useful in future robotics projects. To achieve this goal, the simplified league factors out the most complex aspects of the SSL, namely vision processing and communication, and provides a common architecture for robot development. Further details of the E-League architecture are provided in the next sections of this paper.

We have chosen the RoboCup Small-Size League as our model. Given this, we have identified two major stumbling blocks for teams entering the SSL: *vision* and *communication*. For the E-League, our idea is to provide a standard solution for these two aspects and have teams build the rest[1]. Figure 1 illustrates the current model for the league.

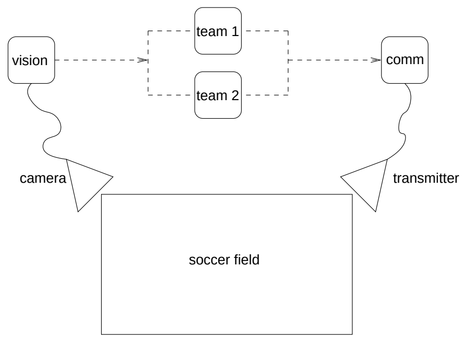

Figure 1: High-level architecture of the E-League.

As shown in Figure 1, the vision and communication components are standard solutions defined and provided by the league. This simplifies practical issues of having to transport and set up cameras as well as the significant problem of coordinating frequencies for multiple communication systems. The next sections detail our plans for the league's vision and communication.

Vision

The E-League uses a standard vision software package to make it easier for teams to enter the league and to speed up the competition. Once the vision software is installed, calibrated and running, it is easy to have multiple playing fields and teams can quickly move from one playing field to another. This would not be possible otherwise, since it takes even experienced SSL teams three to five hours to set up their system on a new playing field. Without the need to move, set up and calibrate cameras, we can have more teams and more games.

In the first year, the league organizers chose to use the *Doraemon* video server package developed by Jacky Baltes[2] (Baltes 2002). This software is open source software released under the Gnu Public License. The Doraemon video server has been in development for over four years and has been used by several robotic teams in international competitions, including RoboCup. Between 1999 and 2002, Doraemon was used by the University of Auckland Small-Size team and was improved and updated each year. It has also been used for two years in the Singapore robotic games.

The cheapest and most practical solution for video capture is to connect a camcorder or surveillance camera with Composite or S-Video out to a Conexant-based frame grabber card [3], e.g., a Hauppauge WinTV PCI board[4]. Note that PVR devices are not compatible with Video4Linux, so these should not be used. The most straightforward option is to use a digital video decoder built on the bt878 chip that runs with a bttv driver[5].

Doraemon includes real-time camera calibration, color calibration and object tracking components. Doraemon also has the ability to calibrate the geometry of the scene from any view, meaning that it is not necessary to have the camera mounted directly overhead relative to the playing field, nor is it necessary to add a wide-angle lens to the camera. Currently, the system is employed with robots wearing colored "hats" or bar codes, but there are also more sophisticated object recognizers that use only a single colored spot on the robot (Baltes 2002). The developers are currently working on a pattern-recognition process using neural networks that does not require any markers (Baltes & Anderson 2003).

Doraemon transmits the position, orientation and velocity of all objects that were found to all clients listening on Ethernet. The messages are transmitted in ASCII via UDP broadcast in a single package. Each message contains eleven of lines: two lines of header information, one line for ball information and eight lines for robot information. An example is shown in Figure 2.

LineNum	content
0	`9 1073821804 0.0013`
1	`-76.3836 -1820.48 2356.39`
2	`1 ball NoFnd 971.056 840.711 35 0 -2.316 58.146`
3	`0 b0 Found 1185.59 309.187 100 0.059 499.282 285.083`
4	`0 b1 Found 1158.59 308.198 100 0.059 499.282 285.083`
5	`0 b2 Found 1086.95 309.187 100 0.059 499.282 285.083`
6	`0 b3 Found 1185.59 309.187 100 0.059 499.282 285.083`
7	`0 y0 Found 989.95 304.588 100 -0.101 413.528 -1.085`
8	`0 y1 NoFnd 1189.95 304.588 100 -0.101 413.528 -1.085`
9	`0 y2 Found 1189.95 304.588 100 -0.101 413.528 -1.085`
10	`0 y3 Found 1189.95 304.588 100 -0.101 413.528 -1.085`

Figure 2: Sample output message from Doraemon.

The first line of each message contains (a) the number of objects ($NumObj = 9$) that video server is currently tracking; (b) the absolute frame number; and (c) the time difference in seconds between this message and the previous message. A client can use the absolute frame number and time difference value to determine if any frames were dropped by the video server. See line 0 in Figure 2.

The second line of each message contains the coordinates (C_x, C_y, C_z), in millimeters, of the camera with respect to the real-world coordinate system. This is used in distributed vision or stereoscopic vision applications and will not be

[1]Note that we also provide a very simple control program for teams that want to focus primarily on hardware development, which may be the case for electrical and/or mechanical engineering classes.

[2]`http://sourceforge.net/projects/robocup-video`

[3]`http://www.conexant.com`

[4]`http://www.hauppauge.com`

[5]`http://www.bytesex.org/bttv`

used in this league. The example shows that in this case, the camera was mounted 2.3m above the playing field. See line 1 in Figure 2.

Following this package header, there are $NumObj$ lines, one for each object that the video server is tracking (i.e., 9 here). Each object line contains the following information:

- the type of object (0=robot, 1=ball);
- the name of the object;
- whether the object was found in the image or if the video server did not find the object and predicted the positions based on previous motion;
- the x, y, and z coordinates of the object, in millimeters;
- the orientation of the object in radians; and
- the velocity of the object in the x and y directions.

The names used for each of the nine objects are `ball` for the ball, `b0` through `b3` for the four robots on the blue team and `y0` through `y3` for the four robots on the yellow team. See lines 2 through 10 in Figure 2.

This example shows that `ball` was not found and that the best estimate of the video server about the position of the ball is ($x = 971$, $y = 840$, $z = 35$). A ball has no orientation and the video server always gives it an orientation of 0 radians. Note that the height (z-coordinate) of all objects is fixed if only a single camera view is used. For example, the height of the ball is 3.5cm and the height of the robot in the next line (below) is 10cm. The best guess for the velocity of the ball is a vector (dx=-2,dy=58).

This example shows that the robot `b0` was found at position (x=1185, y=309, z=100). The orientation of the robot is about 0 degrees and its motion is given by the vector (dx=499, dy=285).

Communication

The Communications component of the league consists of two paths. One path goes from each team's client program to the Communication Server. We refer to this as the input, or *read*, path. The second path goes from the Communication Server to the robots, and we refer to this as the output, or *write*, path. We have defined protocols for both paths. The Communication Server contains two threads, one for reading messages from clients and one for writing messages to robots.

Input messages are passed along an Ethernet link, connecting each team's computer (labeled team1 and team2 in Figure 1) to the computer where the Communication Server is running (labeled comm in Figure 1). The Comm Server listens for messages from clients on a socket. The clients send ASCII messages of the form:

```
[name]:[msg]\n
```

where `[name]` is the name of the robot (as above, `b0` through `b3` and `y0` through `y3`) and `[msg]` is an 8-bit (one byte) message to be sent to the specified robot, i.e., a number between 0 and 127 (however value 0 is reserved as a NULL command, described below). Thus an example of a complete message would be:

```
y0:123\n.
```

The Comm Server maintains an 8-byte command buffer, one byte per robot. Each time an input message is received, the Comm Server updates the byte corresponding to the robot specified in the message. There is no restriction on the frequency with which a client can send a message to the Comm Server. However, keeping within the spirit of the league, we expect that teams will not flood the Comm Server's input channel. In practice, we may find that some type of restriction on the frequency of message transmission is necessary.

Output messages are transmitted to robots using a standard Infra-Red (IR) transmitter connected to the computer via a serial port or USB port. We have been working with the communication tower that comes with the LEGO Mindstorms kit as well as our own custom built infrared transmitter. The Lego tower with two IR transmitter diodes is powerful enough for small labs, but more transmitter diodes are needed for larger playing fields and for adverse environments, such as the vast, open venues where competitions are typically held. Our custom built interface accommodates several transmitter pods with six or twelve diodes[6]. PCB layouts and firmware are available.

The output thread writes continuously to the serial or USB port — wherever the IR transmitter is connected. The Comm Server transmits messages of the form:

```
[START][command-buffer][CHKSUM]
```

where `[START]` is a one-byte start value (255) and `[CHKSUM]` is a one-byte checksum value (only three bits are used, so the values of the checksum ranges from 0 to 7).

The process for the Comm Server output thread is as follows:

1. Lock the command buffer (to prevent the input thread from making changes to it temporarily).
2. Copy the command buffer to the transmit buffer.
3. Set all 8 bytes in the command buffer to 0 (`NULL` command).
4. Release the lock on the command buffer.
5. Calculate the checksum value.
6. Append the checksum to the end of the transmit buffer.
7. Prepend the start value to the front of the transmit buffer
8. Send the transmit buffer over the IR link.
9. Clear the transmit buffer.

Robot Platform

An E-League team consists of 4 robots. The league does not use one standard robot platform (like, for example, the RoboCup 4-Legged League, which uses the Sony AIBO). However, platform specifications are in place to keep teams on an equal plane regarding the cost and sophistication of the robot hardware. Thus we provide maximum specifications for processor capability, in terms of size (RAM) and speed. This allows the option either to purchase an off-the-shelf robot kit or to build one from basic components. Below

[6] `http://www.cs.umanitoba.ca/~jacky/RoboCup-Uleague/`

we list several popular robotic kits that are within the range we are currently testing.

- *Basic Stamp Board of Education (BOE Bot)*
  BASIC Stamp 2 (Parallax custom PIC 16C57C-20/SS)

speed:	20MHz
memory:	2K
approximate cost:	US$229
programming interface:	PBASIC

  `http://www.parallaxinc.com`

- *Handyboard*
  Motorola 68HC11

speed:	2MHz
memory:	32K
approximate cost:	US$219
programming interface:	Interactive C

  `http://www.handyboard.com`

- *LEGO Mindstorms 2.0*
  Hitachi H8/3293

speed:	10MHz
memory:	32K
approximate cost:	US$199
programming interface:	RCX Code, RoboLab, Not-Quite C (NQC), BrickOS and more

  `http://www.legomindstorms.com`

Course

We have been developing an undergraduate course in introductory robotics which uses the RoboCup E-League as a hands-on environment for experimenting with the topics and concepts discussed in the course. The curriculum covers twelve basic topics in introductory robotics, or embodied agents. Students use the LEGO Mindstorms platform for experimentation and the Not Quite C programming language[7] (Baum 2000; Baum *et al.* 2000).

The course is divided into lecture and lab components. The lecture topics follow from the list itemized below, approximately once per week. If there is extra time in the term, it is interesting to add topics such as Robots and Society, Science Fiction (e.g., “I, Robot” (Asimov 1950)), Cognitive Science (Minsky 1987), Synthetic Psychology (e.g., Braitenberg vehicles (Braitenberg 1984)) and Human-Robot interaction.

- *Introduction to embodied agents.* The course begins with an introduction to robotics and agent-based artificial intelligence (Russell & Norvig 1995). Some of the basics of building with and programming LEGO Mindstorms (Martin 1996; Martin *et al.* 2000) using Not Quite C (NQC) are covered. Lab exercises introduce the basics of NQC programming.
- *Motors, effectors and actuators.* Design and operation of robot actuators are discussed (McKerrow 1991; Martin 2000). This includes the basic types of actuators and how robots are typically grouped according to their implementation (i.e., mobile robots, grasping robots, etc). Different modes of locomotion are described. Degrees of freedom are explained. Lab exercises let students experiment with different types of locomotion, such as wheels versus treads and 2-, 3- and 4-wheeled robots using different drive-train configurations.
- *Sensors and vision.* Design and operation of robot sensors are outlined (McKerrow 1991; Martin 2000). Detailed discussion of robot vision ensues, including in particular, operation of the Doraemon vision server. Basic algorithms for object detection and color calibration are described. Lab exercises give students the opportunity to learn how to calibrate the vision server and experiment with various lighting conditions.
- *Knowledge representation: mapping and memory.* Various methodologies for knowledge representation are discussed (Nilsson 1998). The development of methodologies for memory management (e.g., long-term versus short-term) are outlined, based on the historical AI literature. These general techniques are applied to robotics for handling tasks such as localization and mapping of the environment. Lab exercises include a simple localization task.
- *Control I: deliberative, reactive and hybrid architectures.* The history of robot control is presented (Murphy 2000). Early attempts at deliberative and reactive solutions are described, as well as the development of hybrid architectures. The trade-offs of each type are discussed. In lab, students program their robots using deliberative and reactive architectures.
- *Control II: subsumption (layered) and behavior-based architectures.* The classic Brooks subsumption architecture is presented (Brooks 1986), along with a variety of behavior-based control techniques which followed historically (Arkin 1998; Birk 1998; Mataric 1997). Lab exercises have students constructing a simple subsumption architecture for their robots.
- *Control III: BDI architectures.* The development of agent-based control systems is described (Wooldridge 2002). In particular, Belief-Desire-Intention (BDI) architectures. In the lab, students modify their robots to use a BDI architecture.
- *Robotic Simulators.* The notion of robotic simulators is discussed (Balch 1998; Noda & Stone 2003), both as tools for developing high-level multi robot behaviors as well as low-level individual robot control algorithms. The benefits and drawbacks of using robotic simulators is discussed. The RoboCup Simulator League is described. As a lab exercise, students experiment with the either the RoboCup simulator or a simplified alternative. Note that the use of the RoboCup Simulator can require non-trivial start-up time. Some packages have been developed that ease interfacing to the simulator; these may be used in conjunction with the RoboCup simulator.
- *Learning I: Reinforcement learning.* Learning in robots is introduced (Mataric 1994). The classic reinforcement learning algorithms are outlined (Kaebling, Littman, &

[7] `http://www.baumfamily.org/nqc/index.html`

Moore 1996). Students use the RoboCup simulator to program a simple reinforcement learning task.

- *Learning II: Evolutionary learning.* Evolutionary and co-evolutionary learning techniques are described, in particular genetic algorithms, perceptrons and neural networks (Harvey, Husbands, & Cliff 1992; Watson, Ficici, & Pollack 1999). Their application to robot controllers is presented. Lab exercises involve programming a simple evolutionary algorithm.
- *Multi Robot Societies I: Communication.* Methods of communication between robots are discussed, primarily from a software, agent-based perspective (Wooldridge 2002).
- *Multi Robot Societies II: Artificial Life.* Multi robot behaviors such as swarming are described here (Colorni, Dorigo, & Maniezzo 1992; Balch, Khan, & Veloso 2001). Examples from nature are presented, such as ant behaviors. Team behaviors are also discussed, in particular as they relate to RoboCup soccer.

As indicated above, the lab components employ the techniques discussed in lecture, building week by week toward a RoboCup team that uses the setup described in the first part of this paper. The students can be split into small groups of 2-4 students; and each group can work on building and programming one robot. Another option is to divide the class into larger groups and have each group build an entire team.

Assessment for the course includes exams, weekly reading and lab assignments, and two robot contests using the setup described in the previous section. The first contest, held about mid-way through the term, involves robot “challenges” such as penalty kicks and an obstacle course through the soccer field. The second contest, held at the end of the semester, is a full soccer game. These contests are extremely motivating for the students.

A course like this provides good opportunities for students to improve non-curricular skills such as teamwork, technical writing and oral communication. Students should be encouraged to keep lab notebooks and write up experiments conducted throughout the term. A lab report can be submitted following each contest. RoboCup team description papers can be used as examples. These are found in the proceedings of the RoboCup Symposia, published each year (since 1997) as a Springer Verlag Lecture Notes volume[8]. As part of the contests, students should be encouraged to describe to their classmates their robot hardware and software in oral presentations. Formal, conference-style presentations are very effective. Students can be required to produce slides and prepare concise 10-minute talks. They should be required to listen to each others’ presentations and ask questions at the end.

Summary

We have presented our design for the E-League, a new, entry-level league within RoboCup, providing a place for new undergraduates or entry-level students. The straightforward setup also offers instructors an exciting way to give their students hands-on experiences with embodied agents, to support introductory robotics and/or artificial intelligence curriculum.

The league architecture consists of a common platform for Vision and Communication. Both will be provided by the league organizers at any competition venues. Precise specifications for these platforms will be made available to any teams who wish to participate, so that they may build equivalent setups in their home labs in order to practice and develop their teams. Some of these details are already available as open source software.

The league will be open to interested parties in 2004. Regularly updated information and a discussion board can be found on our web page:

`http://agents.cs.columbia.edu/eleague`

As well, full details of the curriculum described here will become available on this web site.

[8]Up until 2001, these were included in the printed book; subsequently, they are on CDROM.

Acknowledgments

We gratefully acknowledge the contributions of Jonah Tower (Columbia University) and Col David Livingston (Virginia Military Institute) during the initial phase of this project.

References

Anderson, J.; Baltes, J.; Livingston, D.; Sklar, E.; and Tower, J. 2003. Toward an Undergraduate League for RoboCup. In *Proceedings of RoboCup-2003: Robot Soccer World Cup VII*.

Arkin, R. C. 1998. *Behavior-Based Robotics*. MIT Press.

Asimov, I. 1950. *I, Robot*. Garden City, NY: Doubleday.

Balch, T.; Khan, Z.; and Veloso, M. 2001. Automatically Tracking and Analyzing the Behavior of Live Insect Colonies. In *Proc of Agents-01*.

Balch, T. 1998. Robots move: Position paper on simulation. In *Working Notes of AAAI 1998 Spring Symposium*.

Baltes, J., and Anderson, J. 2003. Learning orientation information using neural nets. In *submitted to Robocup-03 Symposium*.

Baltes, J. 2002. Yuefei: Object orientation and id without additional markers. In *RoboCup-01: Robot Soccer World Cup V*. New York: Springer.

Baum, D.; Gasperi, M.; Hempel, R.; and Villa, L. 2000. *Extreme MINDSTORMS: An Advanced Guide to LEGO MINDSTORMS*. Apress.

Baum, D. 2000. *Dave Baum’s Definitive Guide to LEGO Mindstorms*. APress.

Birk, A. 1998. Behavior-based robotics, its scope and its prospects. In *Proc of 24th IEEE Industrial Electronics Society Conf.* IEEE Press.

Braitenberg, V. 1984. *Vehicles: Experiments in Synthetic Psychology*. MIT Press.

Brooks, R. 1986. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation* 2(1).

Colorni, A.; Dorigo, M.; and Maniezzo, V. 1992. An Investigation of some Properties of an Ant Algorithm. In *Proc of PPSN-92*. Elsevier Publishing.

Harvey, I.; Husbands, P.; and Cliff, D. 1992. Issues in Evolutionary Robotics, Cognitive Science Research Paper Serial No. CSRP 219. Technical report, Univ of Sussex, Brighton, UK.

Kaebling, L.; Littman, M.; and Moore, A. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* 4:237–285.

Kitano, H.; Asada, M.; Kuniyoshi, Y.; Noda, I.; and Osawa, E. 1997. Robocup: The robot world cup initiative. *Proceedings of the First International Conference on Autonomous Agents (Agents-97)*.

Martin, F.; Mikhak, B.; Resnick, M.; Silverman, B.; and Berg, R. 2000. To Mindstorms and Beyond: Evolution of a Construction Kit for Magical Machines. *Robots for Kids: Exploring New Technologies for Learning*.

Martin, F. 1996. Kids Learning Engineering Science Using LEGO and the Programmable Brick. In *Proc of AERA-96*.

Martin, F. 2000. *Robotic Explorations: A Hands-On Introduction to Engineering*. Prentice Hall.

Mataric, M. J. 1994. Reward Functions for Accelerated Learning. In Cohen, W. W., and Hirsh, H., eds., *Proc of Machine Learning Conf*. Morgan Kaufmann Publishers.

Mataric, M. J. 1997. Behavior-Based Control: Examples from Navigation, Learning, and Group Behavior. *Journal of Experimental and Theoretical Artificial Intelligence, special issue on Software Architectures for Physical Agents* 9(2-3).

McKerrow, P. J. 1991. *Introduction to Robotics*. Addison-Wesley Publishing Co.

Minsky, M. 1987. *Society of Mind*. Picador.

Murphy, R. 2000. *Introduction to AI Robotics*. MIT Press.

Nilsson, N. J. 1998. *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann.

Noda, I., and Stone, P. 2003. The RoboCup soccer server and CMUnited clients: Implemented infrastructure for MAS research. *Autonomous Agents and Multi-Agent Systems* 7(1&2). To appear.

Russell, S. J., and Norvig, P. 1995. *Artificial Intelligence: a Modern Approach*. Prentice Hall.

Sklar, E.; Eguchi, A.; and Johnson, J. 2002. RoboCupJunior: learning with educational robotics. In *Proceedings of RoboCup-2002: Robot Soccer World Cup VI*.

Watson, R. A.; Ficici, S. G.; and Pollack, J. B. 1999. Embodied Evolution: Embodying an Evolutionary Algorithm in a Population of Robots. In *Proc of CEC-99*.

Wooldridge, M. 2002. *MultiAgent Systems*. Wiley.