An Embodied Evolutionary System to Control a Population of Mobile Robots using Genetic Programming

Anderson Luiz Fernandes Perez[1], Guilherme Bittencourt[1], and Mauro Roisenberg[2]

[1] Department of Automation and Systems
Federal University of Santa Catarina - UFSC
Florianópolis , SC, Brazil
{anderson,gb}das.ufsc.br

[2] Department of Computer Science
Federal University of Santa Catarina - UFSC
Florianópolis , SC, Brazil
mauro@inf.ufsc.br

Abstract. In this paper, an embodied evolutionary system, able to control a population of mobile robots, is proposed. This system should be able to execute tasks such as collision-free navigation, box pushing and predator and prey. The proposed system has the following characteristics: i) it extends the traditional genetic programming algorithm to allow the evolution in a population of physical robots; ii) the evolutionary process occurs in an asynchronously way among the robots in the population; iii) it is fail-safe, because it allows the continuation of the evolutionary process even if only one robot remains in the population of robots; iv) it saves the information about the more adapted individuals in a kind of memory; v) it has an execution and management environment that is independent of the evolutionary process.

Key words: Evolutionary Robotic, Distributed Genetic Programming, Embodied Evolutionary System.

1 Introduction

Programming a robot to execute a certain task often demands that the programmer has a good knowledge on the problem domain, i.e., the programmer is responsible for describing all the necessary steps that the robot should take to execute the task.

Often the programmer does not have the means to predict possible problems the robot could face, especially if the environment in which the robot is acting is dynamic or non-structured. In this case, it is important that the robot has a high level of autonomy and is given mechanisms that allow its self-adaptation, so it can make decisions in situations for which it was not programmed.

To make the control system of a robot more dynamic, that is, adaptable, it is necessary to use some development technique that allows the system to modify itself along its execution. Evolutionary Computation (EC) [1], through its Evolutionary Algorithms [2], allows the development of adaptable control systems for mobile robots [3].

Evolutionary Robotics (ER) [4] [5] aims at the development of adaptable control systems, based on the EC techniques. One of the research areas in ER is Embodied Evolution (EE) [6], that unites ER to cooperative robotics [7], [8]. In EE, the evolutionary process takes place among the robots, that is, the reproduction takes place among the individuals that are part of the robot population.

Genetic Programming (GP) is an EC technique that aims at automatically generating computer programs [9]. The main goal of GP is teaching computers to program themselves, i.e., from specified primary behaviors, the computer must be able to generate a program that satisfies some conditions that aim at the solution of some task or problem.

In this paper, we describe an Evolutionary Control System (ECS) for a population of mobile robots. The ECS is made of two main parts: DGP (Distributed Genetic Programming) and EMSS (Execution, Management and Supervision System). The paper is organized in the following way: Section 2 presents the description of the Evolutionary Control System; Sections 3 and 4 show details of the DGP and EMSS mechanism, respectively; Section 5 describes some problems chosen to evaluate the proposed system; the last section presents the conclusions.

2 Evolutionary Control System

The proposed Evolutionary Control System (ECS) is based on the Genetic Programming (GP) [9] algorithm. The goal is to build, by using only GP, a control system for a population of mobile robots in which the robots interact among them to execute some task. The ECS is made of two main modules. The first one, called DGP (Distributed Genetic Programming) is the algorithm responsible for all the evolutionary process of the robots control system. The DGP is an extension of the traditional GP algorithm to support the control system evolution for the robots that are part of the mobile robots population. The second module, called EMSS (Execution, Management and Supervision System) is responsible for executing and managing the DGP.

Figure 1 illustrates, generically, the functioning of the proposed ECS. In each robot, the EMSS, responsible for managing the evolutionary control system, is executed. Each robot has a local population[3] that interacts with the other robots local population. In each generation, parts (subtrees) of the best individual in each robot are sent to all the other robots.

The advantage in using GP to develop control systems for mobile robots is that the handled structures (functions and terminals) are high level, which may

[3] The local population is the set of programs that may solve a problem. These programs are embedded on the robot. A problem is a task that the robot should execute, for instance, collision-free navigation.

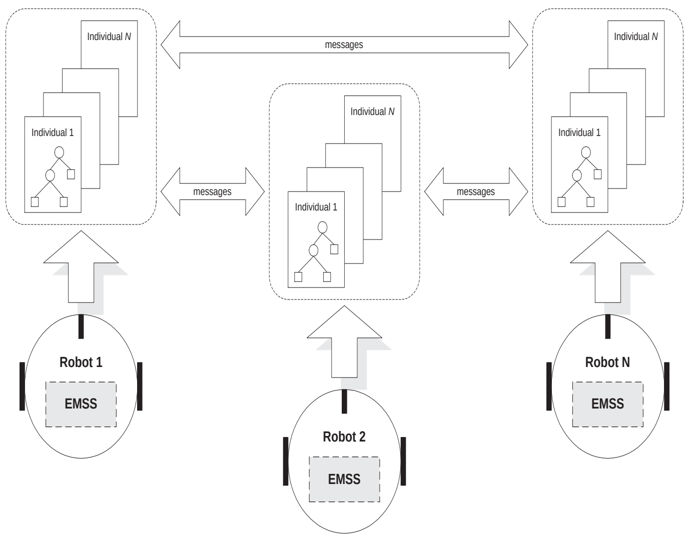

Fig. 1. Schematic Representation of ECS.

cause a better performance in the evolutionary process. The search space tends to be smaller, when compared to other EC techniques that handle lower level structures, such as Genetic Algorithms (GA), for instance.

A disadvantage of GP as to other EC techniques is that the definition process of the terminals and especially of the functions requires more attention and experience from the programmer. Functions and terminals developed for a particular problem may not apply to other kinds of problems. Therefore, the set of functions and terminals must be designed to be as comprehensive as possible, that is, with few modifications or even none, in such a way that it is possible to use the same functions and terminals in many kinds of problems, only redefining the evaluation function.

Sections 3 and 4 describe in details how DGP and EMSS work, respectively.

3 Distributed Genetic Programming - DGP

DGP (Distributed Genetic Programming) is an extension of the traditional GP algorithm. The DGP is based on Microbial GA [10], a GA variation. Its functioning similar is to the genetic combination (infection) that takes place in bacteria, where DNA segments are transferred between two members of the population.

In DGP two population sets are considered. The first, called local set or $P_{local_{R_i}}$, refers to the local population of each R_i robot, i.e., the set of individuals or solutions that are embedded on the robot. Each $x \in P_{local_{R_i}}$ represents a candidate solution to a problem. The second set, called total set or P_{total}, is made of the union of all the local populations in each robot, where: $P_{total} = P_{local_{R_1}} \cup P_{local_{R_2}} \cup \cdots \cup P_{local_{R_n}}$. The evolutionary process takes place always considering the total population, that is, parts (subtrees) of a local individual of a certain robot may be considered in another robot's local population evolutionary process.

Differently from other approaches in EE, in DGP the evolutionary process is asynchronous, that is, it is not necessary that two robots are synchronized to reproduce. In DGP, parts of a more adapted individual are sent to all the other robots. The step sequence in DGP is the following:

1. Randomly create a program population;
2. Iteratively execute the following steps until some halt criterion is satisfied:
   (a) Evaluate each population program through a heuristic function (fitness) that expresses its fitness, that is, how close to the ideal solution is the program;
   (b) Receive[4] parts of a remote[5] individual sent by another robot;
   (c) Select the t best individuals in the local population, using the tournament selection method;
   (d) Randomly select a part of the best (more adapted) local individual and broadcast it to the other robots;
   (e) Compare if the worst locally selected individual's fitness is lower than the remote individual's fitness. If so, execute the mutation operator, replacing parts of the individual by the parts received from a remote individual;
   (f) Execute the crossover and mutation operators;
3. Return the best program found.

The selection method used in the DGP is tournament selection with the maintenance of the parents after the crossing. This is an elitist technique known in the GP's literature as steady-state genetic programming. The remotely received parts are added to the worst individual's tree, from the t selected best, following equation 1:

$$M(A) = \left\{ \begin{array}{ll} Muta(A) & \text{if } \mathit{FitnessR} > \mathit{FitnessL} \\ A & \text{if } \mathit{FitnessR} \leq \mathit{FitnessL} \end{array} \right\} \tag{1}$$

where *FitnessR* is the value of the remote individual's fitness, which sent a part of its tree. *Muta(A)* is the mutation function, where a part of the A tree in the local individual is randomly chosen to be replaced by the tree part of the remote individual.

The messages passing between the robots must contain fitness and a part of the tree (subtree) of the local population individual. For such, all the functions and terminals receive an unique numerical identification, an even number for

[4] In each execution cycle the DGP considers only one received message. Each new message, containing parts of a remote individual, is stored in the local buffer. The buffer is overwritten every time a new message if received.

[5] A remote individual is a program that is part of the local population of another robot.

each function and an odd number to each terminal. For instance, in the foraging task, in which a set of robots must navigate through an environment searching for food, the set of functions and terminals could be defined according to Table 1.

Table 1. Functions and Terminals sets

Functions			
Name	**Arity**	**Id.**	**Definition**
FoodAhead	2	2	If found food, perform left node; otherwise, perform right node.
Prog2	2	4	Perform two branches of the tree.
Prog3	3	6	Perform three branches of the tree.
Terminals			
TurnRight	0	1	Turn right (15 degrees).
TurnLeft	0	3	Turn left (15 degrees).
GoForward	0	5	Go forward (300 ms).
Return	0	7	Return (300ms).

In Table 1, the Id. column represents the identification of functions and terminals. Figure 2 illustrates an example of the DGP functioning.

In Figure 2(a) robot 1 is sending part of its tree to robot 2 (Figure 2(b)). In this example, the local population individual of the robot, which is being used, has a fitness value of 15. And the robot 2 individual has a fitness value of 10. Randomly, part of the robot 1 individual's tree is sent to robot 2. The message sent is formed by the following elements: $\mathbf{M = \{15,2,3,7\}}$. That is, the fitness value, *FoodAhead*, *TurnLeft* and *Return*. After comparing the fitness values, part of the robot 2 individual's tree, which is also randomly picked, is replaced by the part of the robot 1 tree (Figure 2 (c)).

It is important to emphasize that for a correct functioning of DGP all robots must contain the same sets of functions and terminals, or at least they must use the same functions and terminals for a particular problem. Differently from other approaches in EE, such as [7], [11], [12], DGP assures the continuity of the control system of the evolutionary process, even when there is a problem with the other robots that are part of the robot population. This is possible because each robot has a local population of programs, which guarantee the continuity of the evolutionary process.

The main difference between classical GP algorithm and DGP is the size of population. Classical GP algorithm needs a population with a great number of

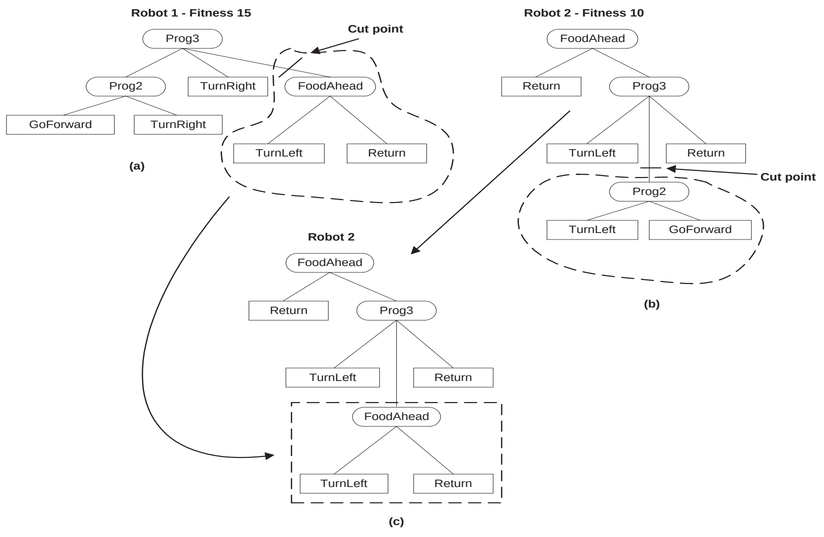

Fig. 2. DGP's Example.

individuals to converge for a solution. In DGP the population in each robot is small and the evolutionary process is faster than GP. However DGP algorithm can never converge for a solution if other robots crash.

4 Execution, Management and Supervision System - EMSS

The EMSS (Execution, Management and Supervision System) is the system responsible for managing the evolutionary process that takes place in an embedded fashion in each robot. Besides DGP, EMSS has other components necessary to the correct functioning of each robot's control system. Figure 3 illustrates the EMSS components and the relationships among them.

Below is the complete description of the goal and functioning of each component that is part of the EMSS and the connection among them (when there is one).

Evolutionary Control (EC): it is the main component in the EMSS, in which the DGP is implemented. The EC works as an interpreter and is responsible for creating, randomly, a local individual population through the functions and terminals library. In each new generation, the generated individuals are tested and executed by the EC. The ADF (Automatically Defined Functions) [13] are also managed by this component. The EC is linked to the Communication Manager, to the Library Manager and the Memory.

Memory: the memory goal is to store the representation of the most adapted individuals in each generation. For instance, considering the example described in Section 3, the best individual in Robot 1, according to the representation rules

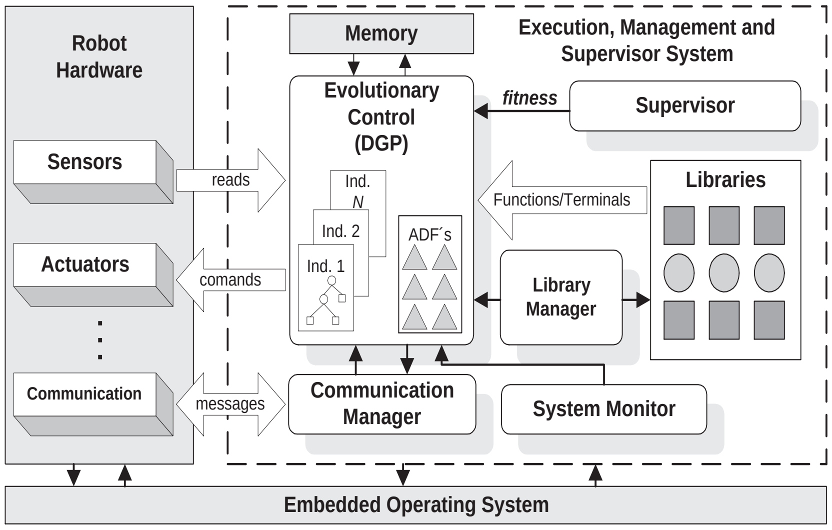

Fig. 3. Basic Structure of EMSS.

for functions and terminals, could be represented as the following: **(6, 4, 5, 1, 1, 2, 3, 7)**, which represents the following structure as to functions and terminals: *(Prog3, Prog2, GoForward, TurnRight, FoodAhead, TurnLeft, Return)*. The **(6, 4, 5, 1, 1, 2, 3, 7)** representation, plus the fitness value are stored in the memory so they can be used in a retrieval process, in case the EC fails or even to optimize tasks that involve more than one competence, such as avoid obstacles and move an object from one place to another. In this case, the memory works as a kind of system photograph, which can be reused to speed up the learning process through experiences taken place in the past.

Communication Manager (CM): it is the component responsible for sending and receiving the messages passing among the robots. In the message sending process, the EC sends the CM the sequence of functions and terminals and the fitness value that will be sent to the other robots. The CM creates a message containing this information and sends it to the other robots. In receiving, the CM receives the messages sent by the other robots and sends them to the EC.

Library Manager (LM): this component manages the sets of terminals and functions in each robot. In this component all the functions and terminals are identified through an unique identifier, an even number for the functions and an odd number for the terminals, so they can be sent to other robots through the CM. Having the libraries separated from the evolutionary systems represents an additional advantage, since it is possible at any time to add or remove functions or terminals without having the need to redefine the control system. For example, in case the robot receives a set of new sensors with new functions, one only needs to add that information to the terminal set in the EMSS libraries. That makes the individuals generated by the EC adaptable to the change of characteristics in the robot hardware (morphology).

Supervisor: it is responsible for evaluating and attributing a fitness value to each individual, that being done through a punishment and reward method. For this method to work properly, for each task the robot has to do one must define how and when reward and punishment take place. For example, if the task is collision-free navigation, reward and punishment can be defined according to the number of robot's collisions. At each bumped with an obstacle, the fitness value is decreased (punishment), otherwise, the value is incremented from time to time (reward).

Monitor: it is the component responsible for evaluating the system execution. In case the system is inactive, i.e., the robot stays still for a long period of time, the monitor activates an EC reinitializing process. When the EC is reinitialized, the image of the best individual, which is stored in memory, is retrieved and the evolutionary process initiates from this individual, that is, the local population is completed through the use of parts of that individual's tree, instead of starting from a random population, as it happens in the original GP algorithm.

The EMSS can be implemented directly on the robot hardware or as a task running on an operating system, such as CubeOS [14], Arena [15] and Robios [18]. The advantage of running on an operating system is being able to use its functionalities, such as concurrent control, memory management and hardware management.

5 System Evaluation

With the purpose of evaluating the proposed control system, we intend to make some experiments, in both simulator and real robots, in three different problems in mobile robotics.

Collision-free Navigation: in this kind of problem, the robots must wander through an environment containing various obstacles in different sizes and shapes. The goal for each robot is to detect and avoid the obstacles, and also the other robots.

Box Pushing: this is a typical cooperation problem, in which a group of robots must push a box from one place to another in the environment. Each robot should, first, locate and approach the box, and then, with the help from the other robots (cooperation), move the box to another position. This problem may be implemented in two different ways. In the first one, considering that there are only the robots and the box in the environment. In the second one, considering that besides the robots and the box, there are also in the environment some obstacles that the robots should avoid.

Predator and Prey: in this kind of problem there are two distinct classes of robots. The first one is made of the predators, which have as their primary goal to "attack" the prey. The second one is made of the prey that must avoid being captured by the predator. One robot class behavior affects the other's behavior (co-evolution), because the prey's control system adaptation to run causes the predator's control system adaptation to hunt, and vice versa.

The three problems described above were chosen for the evaluation experiments because they are well disseminated in the mobile robotics area. For such, there are different approaches to solve those problems, such as the ones described by [11], [16] and [17]. The goal is to compare the results obtained with the proposed system against the results obtained with other approaches.

The ECS proposed will be evaluated in both simulated environment and real robots. The Eyebot [18] robots will be used for the experiments. Eyebot is a kind of mobile robot developed to be used in robot soccer. With the experiments described above we intend to validate the message passing mechanism among the robots and to evaluate how easy it is to change the task of robots without reprogrammed whole control system.

6 Conclusion

In this paper, we described the proposal for an Embodied Evolutionary System, based on the Genetic Programming algorithm, to control a mobile robots population.

The proposed system has the following features: (i) it extends the traditional GP algorithm to support evolution in a real robots population; (ii) the evolutionary process takes place in an asynchronous way and continues even when there is a problem with the other robots in the mobile robots population, because each robot has its own local program population; (iii) the use of memory to store information about the tree structure of the most adapted individuals provides a failure retrieval mechanism and allows the maximization of the evolutionary process in tasks that demand more than one competence; (iv) the execution and management environment is independent from the evolutionary process.

The described system is in development. The primitive behaviors, such as *FoodAhead*, *TurnRight*, *TurnLeft*, *GoForward* and *Return* were already implemented in the Eyebot mobile robots from the Department of Automation and Systems - UFSC.

Acknowledgements

The author, Anderson Luiz Fernandes Perez, acknowledge the CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico), Brazil, for the scholarship.

References

1. Fogel, D.B.: What is Evolutionary Computation? IEEE Spectrum **37:2** (2000) 28–32.
2. Whitley, Darrell: An overview of evolutionary algorithms: practical issues and common pitfalls. Information and Software Technology **43:14** (2001) 817–831.

3. Pollack, Jordan B. and Lipson, Hod and Ficci, Sevan and Funes, Pablo and Hornby, Greg and Watson, Richard A.: Evolutionary Techniques in Physical Robotics. ICES (2000) 175–186.
4. Nolfi, Stefano and Floreano, Dario: Evolutionary Robotics. The Biology, Intelligence, and Technology of Self-Organizing Machines. MIT Press (2001).
5. Nolfi, S. and Floreano, D.: Synthesis of autonomous robots through evolution. Trends in Cognitive Science **6:1** (2002) 31–36.
6. Watson, R. and Ficici, S. and Pollack, J.: Embodied Evolution: Distributing an Evolutionary Algorithm in a Population of Robots. Robotics and Autonomous Systems **39:1** (2002) 1–18.
7. Ficici, S. and Watson, R. and Pollack, J.: Embodied Evolution: A Response to Challenges in Evolutionary Robotics. J. L. Wyatt and J. Demiris, Eighth European Workshop on Learning Robots (1999) 14–22.
8. Cao, Yu Uny and Fukunaga, Alex S. and Kahng, Andrew B.: Cooperative Mobile Robotics: Antecedents and Directions. Autonomous Robots **4** (1997) 7–27.
9. Koza, John R.: Genetic Programming: A Paradigm for Genetically Breeding Computer Population of Computer Programs to Solve Problems. MIT Press (1992).
10. Harvey, Inman: Artificial Evolution: A Continuing SAGA In Evolutionary Robotics: From Intelligent Robots to Artificial Life. 8th International Symposium on Evolutionary Robotics (ER2001). Springer-Verlag Lecture Notes in Computer Science LNCS 2217 (2001).
11. Simões, Eduardo D. V. and Dimond, Keith R.: Embedding a Distributed Evolutionary System into Population of Autonomous Mobile Robots. Proceeding of The 2001 IEEE System, Man, and Cybernetics Conference (2001).
12. Nordin, Peter Nordin and Banzhaf, Wolfgang: An on-line method to evolve behavior and to control a miniature robot in real time with genetic programming. Adaptive Behaviour **5:2** (1997) 107–140.
13. Koza, John R.: Scalable learning in genetic programming using automatic function definition Source. Advances in Genetic Programming (1994) 99-117.
14. Kenn, Holger Kenn: CubeOS: A Component-based operating system for autonomous system. Vrije University, Brussels, August (2001).
15. Kingsbury, S. and Mayes, K. and Warboys, B.: Real-time arena: A user-level operating system for co-operating robots. The Interantional Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA) (1998) 1844–1850.
16. Floreano, Dario and Nolfi, Stefano: Adaptive Behavior in Competing Co-Evolving Species. Fourth European Conference on Artificial Life. The MIT Press, Cambridge, MA, Phil Husbands and Inman Harvey (1997) 378–387.
17. Zhang, Byoung-Tak and Cho, Dong-Yeon: Evolving Complex Group Behaviors Using Genetic Programming with Fitness Switching. Artificial Life and Robotics **4:2** (2000) 103–108.
18. Brunl, Thomas: EyeBot Online Documentation. http://robotics.ee.uwa.edu.au/eyebot/ (2006).