Applied Soft Computing 13 (2013) 2003–2016

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Memory-saving memetic computing for path-following mobile robots

Giovanni Iacca[a], Fabio Caraffini[b,c], Ferrante Neri[b,c,*]

[a] INCAS^3 – Innovation Centre for Advanced Sensors and Sensor Systems, P.O. Box 797, 9400 AT Assen, The Netherlands
[b] Centre for Computational Intelligence, School of Computer Science and Informatics, De Montfort University, The Gateway, Leicester LE1 9BH, England, United Kingdom
[c] University of Jyväskylä, Department of Mathematical Information Technology, P.O. Box 35 (Agora), 40014 Jyväskylä, Finland

ARTICLE INFO

Article history:
Received 24 January 2012
Received in revised form
23 September 2012
Accepted 19 November 2012
Available online 11 December 2012

Keywords:
Computational intelligence optimization
Memetic computing
Real-time optimization
Mobile robotics
Path-following

ABSTRACT

In this paper, a recently proposed single-solution memetic computing optimization method, namely three stage optimization memetic exploration (3SOME), is used to implement a self-tuning PID controller on board of a mobile robot. More specifically, the optimal PID parameters minimizing a measure of the following error on a path-following operation are found, in real-time, during the execution of the control loop. The proposed approach separates the control and the optimization tasks, and uses simple operating system primitives to share data. The system is able to react to modifications of the trajectory, thus endowing the robot with intelligent learning and self-configuration capabilities. A popular commercial robotic tool, i.e. the Lego Mindstorms robot, has been used for testing and implementing this system. Tests have been performed both in simulations and in a real Lego robot. Experimental results show that, compared to other online optimization techniques and to empiric PID tuning procedures, 3SOME guarantees a robust and efficient control behaviour, thus representing a valid alternative for self-tuning control systems.

1. Introduction

Some real-world problems, due to real-time, space, and cost requirements, often impose the solution of an optimization problem within limited computational resources. This situation is typical in mobile robots where the specific application might require that all the computation is embedded within the robot hardware (a computer is not involved in the optimization process). In addition, there are some engineering applications that require the solution of complex optimization problems despite a limited hardware. An example of this class of problems is the space shuttle control. Despite the constant growth of the power in computational devices, space applications are an interesting exception. In order to reduce fault risks, very simple hardware is often used on purpose on space shuttles. This choice allows a high reliability of the computational cores. For example, since over 20 years, National Aeronautics and Space Administration (NASA) employs, within the space shuttles, IBM AP-101S computers, see [1]. These computers constitute an embedded system for performing the control operations. The memory of computational devices is of only 1 Mb, i.e. much less capacious than any modern device. It must be remarked that the computational devices on board of a space shuttle should reliably work without any reboot for months or even for years. Thus, the necessity of having an efficient control notwithstanding the hardware limitations (both memory and computational power) arises.

In these cases, the optimization algorithms should perform the task without a high employment of memory and computational resources. Unfortunately, high performance algorithms are usually fairly complex structures employing a population of candidate solutions and other computationally expensive components such as learning systems or classifiers, see e.g. [2].

In the present paper, a mobile robot path-following application is presented, in which the following controller, namely a proportional-integrative-derivative (PID), is tuned on-line by means of an optimization algorithm. Path-following robots, also called automated guided vehicles (AGVs) [3,4], are mobile robots whose main task is following a generic path, denoted e.g. by means of markers or wires on a surface. Since they increase efficiency and reduce costs, path-following robots are nowadays widely used in industry (see Fig. 1) for moving raw materials, transporting pallets and finished goods, removing scrap, etc. They are becoming increasingly popular also in the health-care industry, e.g. for efficient transportation of linens, trash and medical waste, patient meals, soiled food trays, and surgical case carts, and in many other application domains.

In this study, a test path-following application is implemented on a popular commercial hardware, namely the embedded microcontroller of Lego Mindstorms NXT, see Section 2. The robot

* Corresponding author at: University of Jyväskylä, Department of Mathematical Information Technology, P.O. Box 35 (Agora), 40014 Jyväskylä, Finland.
Tel.: +358 14 260 1211; fax: +358 14 260 1021.
E-mail addresses: giovanniiacca@incas3.eu (G. Iacca), fcaraffini@dmu.ac.uk, fabio.caraffini@jyu.fi (F. Caraffini), fneri@dmu.ac.uk, ferrante.neri@jyu.fi (F. Neri).

1568-4946/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2012.11.039

Fig. 1. An industrial path-following mobile robot.

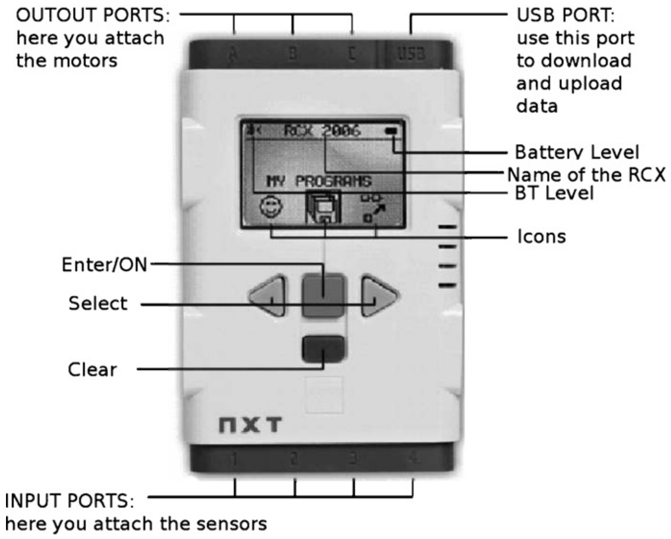
OUTOUT PORTS: here you attach the motors
USB PORT: use this port to download and upload data
RCX 2006
MY PROGRAMS
Battery Level
Name of the RCX
BT Level
Icons
Enter/ON
Select
Clear
NXT
INPUT PORTS: here you attach the sensors

Fig. 2. The NXT brick.

configuration is a two driving-wheels rover, following a black elliptic path over a white background. Two sensors have been used, an ultrasonic sensor for registering the beginning and the end of each iteration of the closed trajectory, and a light sensor to follow the black path. More specifically, the light sensor measures the "amount" of black and white, i.e. the percentage of light, thus allowing a raw measure of the following error. In other words, the path-following task is translated into the requirement that, at each step of the control loop, the light sensor must measure 50% of white and 50% of black during the elliptic path. When the light/darkness proportion changes, an error is measured. The maximum error, obviously, occurs when 100% of white and 0% of black (or dually 100% of black and 0% white) is measured. A global measure of the error along the path is then computed as the integral absolute error (IAE). The optimization problem consists of finding those parameters proportional-integrative-derivative (PID) which allow a minimal IAE: in this way, the robot capable of learning the best set of parameters to follow a generic path. In addition, at every step of the control loop, if the IAE error goes beyond a fixed threshold, a bang-bang control is provided in order to move the robot to the correct place, and another PID parameter set is quickly computed by the optimization algorithm.

It is important to remark that in this work we propose an architecture in which both the control scheme and the optimization algorithm are implemented on board of the Lego Mindstorms, despite its severely limited computational and memory resources, thus avoiding any external computing device. In addition, it should be noted that, due to hardware limitations, the implementation of classical population based algorithms would not be a viable option on an embedded system of this kind. In other words, in this paper we show that the application of an advanced optimization algorithm can allow the accomplishment of a complex industrial task despite the employment of extremely limited hardware conditions.

Although some recent studies proved that population-based algorithms usually have a better performance than algorithms processing a single candidate solution, see [5], still there are some population-less (not only single-solution) methods which are able to provide relatively good results despite a limited memory footprint. If properly designed, population-less algorithms can be competitive with population-based algorithms in specific applications (in accordance with the no free lunch theorem, see [6]) and thus can be a satisfactory alternative when the hardware limitations forbid the use of a complex algorithms. We will refer in these papers to these methods as "memory-saving" algorithms, i.e. algorithms which do not make use of a population of solutions or the support of memory structures. Since this article considers a specific application characterized by severe hardware limitations we will focus on memory-saving algorithms.

The remainder of this paper is structured as follows. Section 2 describes the experimental hardware and software setup. Section 3 introduces the optimization problem and the proposed approach for solving it. More specifically, the real-time tasks which compose the control architecture are described in detail. Section 4 briefly explains the working principles of the optimization algorithm used, namely 3SOME. Section 5 presents the experimental results: a first subsection describes preliminary simulation experiments in which 3SOME was compared with two classical local search algorithms, namely the Hooke–Jeeves and the Nelder–Mead methods, and four different state-of-the-art memory-saving global optimization algorithms. A validation of the simulation results in the real-world case is then presented in the second subsection. Finally, in Section 6 the conclusion of this work is given. For the interested reader, Appendix A presents an additional experimental setup in which two empiric tuning procedures were compared with the 3SOME-based proposed online optimization approach.

2. The mobile robot path follower: hardware and software setup

2.1. Hardware configuration

Lego Mindstorms [7] is a line of the Lego products which includes a programmable NXT brick, electric motors, sensors, and other useful pieces such as gears, axles and pneumatics to build robots or other automated systems. One of the most important features of the Lego Mindstorms is the wide selection of pieces one can choose for designing the robot, which allows many different kinds of robotics applications, such as pick-and-place, fixed and mobile robotics. Originally designed to be used like a toy, the Lego Mindstorms has quickly become an important academic and educational instrument.

What makes the Lego Mindstorms an interesting tool is the NXT brick, see Fig. 2, which includes four input ports for connecting sensors, three output ports for connecting motors, Bluetooth wireless communication, and a reasonably powerful micro-controller, namely a 32 bit ARM7 micro-controller, with 256 KB flash memory