Applying Real-Time Survivability Considerations in Evolutionary Behavior Learning by a Mobile Robot

Wolfgang Freund, Tomás Arredondo V. and César Muñoz
Universidad Técnica Federico Santa María, Departamento de Electrónica
Chile

1. Introduction

In this chapter we investigate real-time extensions for evolutionary mobile robot learning. The learning performance is measured in navigation experiments of complex environments as performed in a Kephera mobile robot simulator (YAKS). All these experiments are done in the context of our recently introduced motivation based interface that provides an intuitive human-robot communications mechanism (Arredondo et al., 2006). This motivation interface has been used in a variety of behavior based navigation and environment recognition tasks (Freund et al., 2006).
Our first heuristic introduces active battery level sensors and recharge zones, which are used as soft deadlines to improve robot behavior for reaching survivability in environment exploration. Based on our previously defined model, we also propose a hard deadline based hybrid controller for a mobile robot, combining behavior-based and mission-oriented control mechanisms.
These methods are implemented and applied in action sequence based environment exploration tasks in a YAKS mobile robot simulator. We validate our techniques with several sets of configuration parameters on different scenarios. We consider soft-deadlines as a dangerous but not critical battery charge level which affect a robot's fitness. Hard-deadlines are considered as a possible (because of partial knowledge) point where, if the robot does not recharge his battery, an unrecoverable final freezing state is possible. Our tests include action sequence based environment exploration tasks. These experiments show a significant improvement in robot responsiveness regarding survivability and environment exploration when using these real-time based methods.
The rest of the chapter is organized as follows. In Section 2 a description of our soft-computing based navigation model is given. In Section 3 real-time extensions of our model are presented. In Section 4 we show the experimental setup and test results. Finally, in Section 5 some conclusions and future work are drawn.

2. Soft-computing Based Robotic Navigation

Navigation and environment recognition is something that is taken for granted by most of us but as we see when observing an infant this is a difficult task fraught with peril for the

Source: Frontiers in Evolutionary Robotics, Book edited by: Hitoshi Iba, ISBN 978-3-902613-19-6, pp. 596, April 2008, I-Tech Education and Publishing, Vienna, Austria

inexperienced navigator. Thanks to its tolerance of imprecision and incomplete data, soft-computing has had much recent success in robotic navigation where other more structured or formal methods have not fared so well (Arkin, 1998; Jang et al., 1997; Konolige et al. 1992; Goodrige, 1997 ; Kem & Woo, 2005). Fuzzy logic has been a mainstay of several such efforts: applying fuzzy rule based method for a hexapod obstacle avoidance (Kem & Woo, 2005), using independent distributed fuzzy agents and weighted vector summation via fuzzy multiplexers for producing drive and steer commands (Goodrige, 1997), neuro-fuzzy controllers for behavior design (Hoffmann, 2000), fuzzy modular motion planning (Al-Khatib & Saade, 2003), fuzzy integration of groups of behaviors (Izumi & Watanabe, 2000), multiple fuzzy agents for behavior fusion (Barberá & Skarmeta, 2002), *GA* based neuro fuzzy reinforcement learning agents (Zhou, 2002), and fuzzy logic integration for robotic navigation in challenging terrain (Seraji & Howard, 2002).

Using fuzzy logic as a motivation toward a variety of useful behaviors is something that has not seen wide usage in robotics previously. Toward this goal, we have implemented such motivations (e.g. a need, desire or want) as fuzzy fitness functions that serve to influence the intensity and direction of robotic behaviors (Arredondo et al., 2006). In general, motivations are accepted as involved in the performance of learned behaviors given that a learned behavior may not occur unless it's driven by a motivation (Huitt, 2001). Having a variety of motivations helps produce diverse behaviors some of which have a high degree of benefit (or fitness) for the organism.

In our experiments, we have used fuzzy membership functions (Fig. l) toward implementing a motivation based interface for determining robotic fitness. There are five triangular functions used for each of the four motivations in our experiment (Very Low, Low, Medium, High, Very High).

The motivation set (M) considered in this study includes: curiosity (m_1), homing (m_2), and energy (m_3). These motivations are used as input settings (between 0 and 1) prior to running each experiment.

During training, a run environment (room) is selected and the GA initial robot population is randomly generated. After this, each robot in the population performs its navigation task and a set of fitness values corresponding to the performed task are obtained (f_1 through f_3). Finally, robotic fitness is calculated using the fitness values information provided by the simulator and the different motivations at the time of exploration (Fig. 2).

Takagi-Sugeno-Kang (TSK) fuzzy logic model is used, TSK fuzzy logic does not require deffuzification as each rule has a crisp output that is aggregated as a weighted average (Jang et al., 1997).

The membership functions used are given in Fig. 1. Sample fuzzy rules (numbers 9 and 10) are given as follows:

if (f_1 == M) and (f_2 == V.L.) and (f_3 == V.L.) then
    $f[9] = m_1 f_1 C[3]+m_2 f_2 C[1]+m_3 f_3 C[1]$

if (f_1 == M) and f_2 == V.L.) and (f_3 == L) then
    $f[10] = m_1 f_1 C[3]+m_2 f_2 C[1]+m_3 f_3 C[1]$

The values for these fitness criteria are normalized (range from 0 to 1). The criteria and the variables that correspond to them are: amount of area explored (f_1), proper action termination and escape from original neighborhood area (f_2), and percent of battery usage (f_3). These fitness values are calculated after the robot completes each run. The f_1 value is

determined by considering the percentage area explored relative to the optimum, f_2 is determined by $f_2 = 1 - l/L$, where l is the final distance to robot's home and L the theoretical maximum value. Finally f_3 is the estimated total energy consumption of the robot considering each step.
The final fuzzy motivation fitness value (F) is calculated using TSK based fuzzy logic (three fuzzy variables with five membership functions each: 3^5 = 243 different fuzzy rules) as shown in Fig. 3 and using the membership functions to compute μ, values. For the coefficient array C we used a linear function.

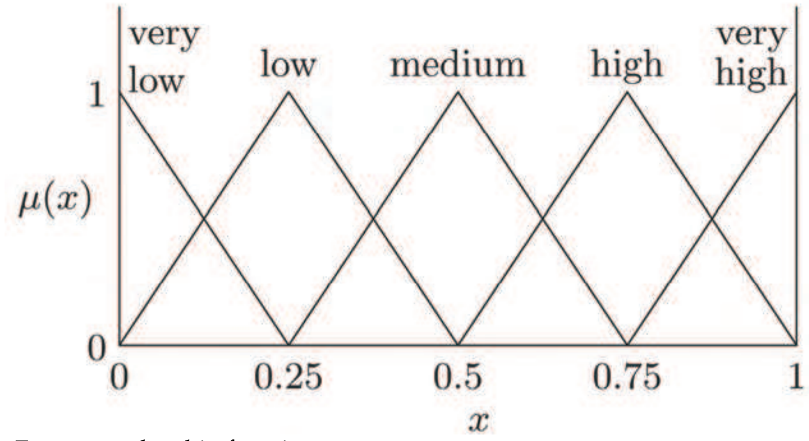


Figure 1. Fuzzy membership functions

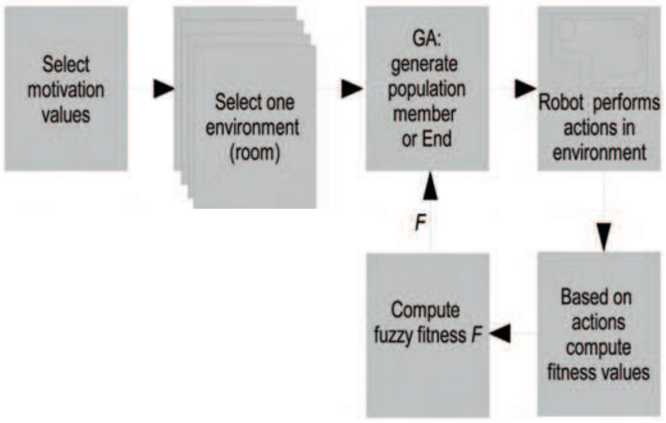

Select motivation values
Select one environment (room)
GA: generate population member or End
Robot performs actions in environment
Based on actions compute fitness values
Compute fuzzy fitness F
F

Figure 2. System overview

```
Algorithm FuzzyFitness
Input:
   $N$ : number of fuzzy motivations;
   $M$ : number of membership functions per motivation;
   $X[N]$ : array of motivation values preset;
   $Y[N]$ : array of fitness values;
   $C[N]$ : array of coefficients;
   $\mu[N][M]$ : matrix of membership values for each motivation;
Variables:
   $w[n]$ : the weight for each fuzzy rule being evaluated;
   $f[n]$ : the estimated fitness;
   $n, x_0, x_1, \ldots, x_N$ : integers;
Output:
   $F$ : the fuzzy fitness value calculated;
begin
   $n := 1$;
   for each $x_1, x_2, \ldots, x_N := 1$ step 1 until $M$ do
   begin
      $w[n] := \min\{\mu[1][x_1], \mu[2][x_2], \ldots, \mu[N][x_N]\}$;
      $f[n] := \sum_{i=1}^{N} X[i]Y[i]C[x_i]$;
      $n := n + 1$;
   end;
   $F := (\sum_{i=1}^{N^M} w[i]f[i]) / (\sum_{i=1}^{N^M} w[i])$;
end;
```

Figure 3. Fuzzy fitness algorithm

2.1 Implementation

The YAKS (Yet Another Khepera Simulator) simulator is the base for our implementation. YAKS is a simple open source behavior-based simulator (YAKS) that uses neural networks and genetic algorithms to provide a navigation environment for a Khepera robot as seen in Fig. 5. Sensors are directly provided into a multilayer neural network in order to drive left and right wheel motors (Fig. 4). A simple genetic algorithm is used with 200 members, 100 generations, mutation of 1%, and elite reproduction. Random noise (5%) is injected into sensors to improve realism. The GA provides with a mechanism for updating neural network weights used by each robot in the population that is being optimized. An overview of our fuzzy fitness implementation is shown in Fig. 6.

Outputs of the Neural Network are real valued motor commands (Motor_L and Motor_R) between 0 and 1 which are discretized into one of four actions (left 30°, right 30°, turn 180°, go straight). This follows the Action-based environmental modeling (AEM) search space reduction paradigm (Yamada, 2005).

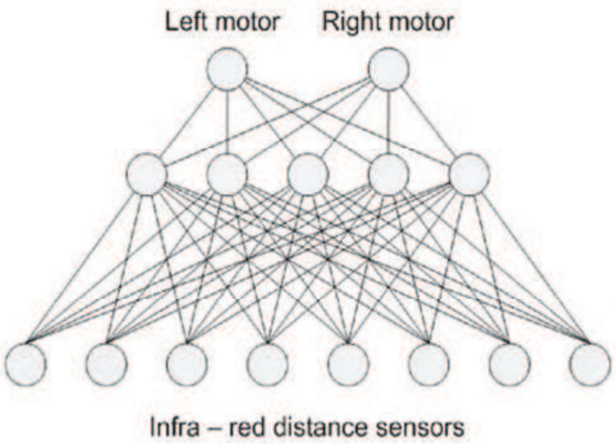


Figure 4. Robotic neural network

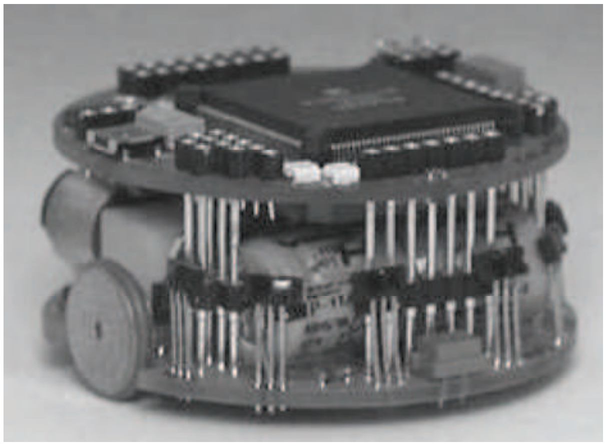


Figure 5. Kephera robot

3. Real-Time Extensions

Real-time systems are concerned with real-world applications, where temporal constraints are part of system specification imposed by the environment, i.e. firm-deadlines in QOS environments, soft-deadlines in non-critical control applications and hard deadlines in safety-critical systems. In the last years more research effort have been made applying soft-computing techniques to real-time control problems (Wang & Lee, 2003; Jha et al., 2005; Seraji & Howard, 2002; Maione & Naso, 2003; Ziemke et al., 2005). The main advantage over traditional control mechanisms is in the additional robustness regarding lack or poor environmental information (if not the problem definition itself) which concerns almost all real-time control applications (Sick et al., 1998).

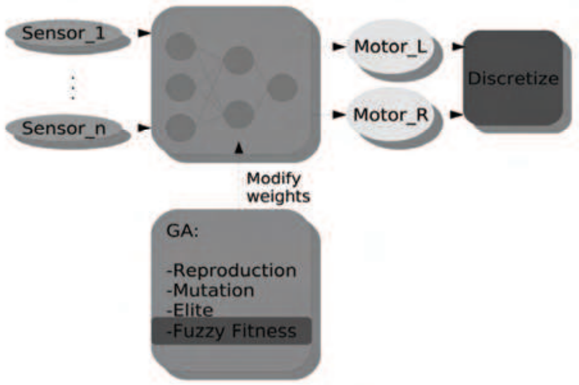


Figure 6. Robotic system implementation

On the other hand, soft-computing based methods are more intuitive than strict formal models, soft-computing (e.g. fuzzy logic) aims to gain from operator perceptions and through iterative improvements in the associated rule set tries to obtain the capabilities of the real expert. However, not much attention has been given to real-time considerations, regarding soft or hard deadlines. Some important aspects of real-time must be taken into account: how could soft-computing techniques, such as fuzzy logic, neural networks or genetic algorithms affect systems responsiveness and survivability?

In order to introduce our behavior based mobile robot methodology in a real-world application, we introduce an active battery sensor to allow for the detection of low battery conditions and we also provide various number of recharge zones within different room configurations. These real-time extensions must be capable of supporting different sets of motivations while also improving survivability and exploration performance.

Our primary goal consists of full environment exploration considering energy consumption and recharge zones. To reach this objective, the robot's behavior must be influenced through periodic energy evaluation for recharging the battery before it is be too late. In this approach we consider soft-deadlines as a dangerous but not critical battery charge level which affect robot's fitness. Hard-deadlines are considered as a possible (because of partial knowledge) point where, if the robot does not recharge his battery, an unrecoverable final freezing state is possible. Soft vs hard-deadlines force a change in the robot's operation from behavior-based to mission-oriented (hybrid), which guides the robot using the shortest known path to a nearest previously found charging zone.

During environment exploration, autonomous or semi-autonomous mobile robots are confronted with events which could be predictable such as walls and static objects, or unpredictable such as moving objects or environmental changes. Some of these events must be attended in real-time (responsiveness) to guarantee the robot's integrity (survivability) (Kopetz, 1997).

Traditional control mechanisms are based on reliable real-time systems, i.e. time constraints over executions and predictability (Gheith & Schwan, 1993), also known as dependable systems (Motet & Geffroy, 2003), e.g. the mars pathfinder or DUSAUV, a semi-autonomous underwater vehicle presented in (Li et al., 2005). On the other hand, soft-computing based methods have not been widely used in this arena due to their inherent uncertainty.

In order to introduce real-time considerations into our behavior-based mobile robot for a real-world application, we extend our model by using temporal constraints during the navigation test-phase. The constraints considered include energy consumption and finite battery charge capacity.

In our approach soft-deadlines dynamically affects the robots navigation. This could influence behaviors to avoid highly energy consuming actions and could guide the robot's movement to a recharging zone as necessary.

When a critical battery level is reached, the previously defined method is no longer useful. A responsive real-time method is needed to, if possible, guarantee survivability (Kopetz, 1997). Strictly speaking, we can't guarantee survivability because of the robots partial knowledge of the world map which, initially, has no recharge zones mapped (we do not consider the starting point as a recharging zone). Nevertheless, because of the off-line robot training-phase, we expect that the trained robot (e.g. NN) will be capable of finding charging zones. Using the charge zone information obtained on-line, the robot applies real-time based navigation. We establish a hard-deadline as the point of the robot's unrecoverable final freezing state. Before reaching this deadline (with a 10% safety margin) the robot's operation mode changes from behavior-based to mission oriented, following the shortest path to the nearest previously found charging zone (Tompkins et al., 2006).

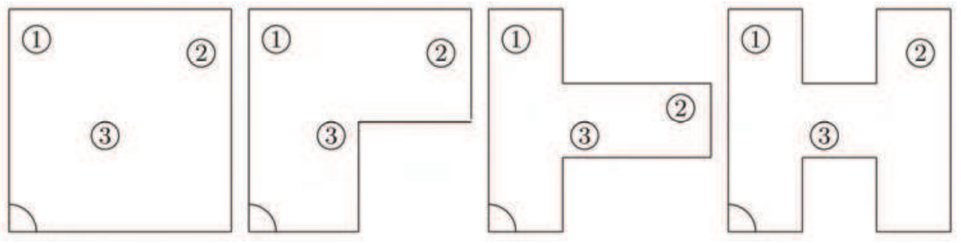
4. Experimental Evaluation

The major purpose of the experiments reported in this section is to study the influence of our real-time extensions over the robot's behavior, considering survivability and exploration capability.

We have designed four different rooms (environments) for the robot to navigate in. We denote these rooms as: S-ROOM (the simplest), L-ROOM, T-ROOM and H-ROOM (most complex). Walls are represented by lines and we designate up to three charging zones (see circles in Fig. 7). The starting zones for each room will be the lower left corner (quarter circles in Fig. 7).

We will denote by NRT the behavior-based algorithm which operates the robot without any real-time considerations, i.e. the battery level has no influence over robot's behavior but, if it comes near to a charging zone the battery level is updated to his maximum capacity. The main characteristics of NRT are:

- the battery level has no influence on the robot during training phase and,
- there is no input neuron connected to the battery sensor.



a) S-ROOM b) L-ROOM c) T-ROOM d) H-ROOM

Figure 7. Experiment rooms layout with starting and recharging zones

We denote by SRT the algorithm which operates the robot with soft-real time considerations, influencing his behavior to avoid a dangerous battery level. This algorithm differs from NRT mainly by

- battery level influences robot's fitness evaluation used by the GA and,
- a new input neuron is connected to a battery level sensor.

Finally, we denote by HRT the hybrid algorithm which operates the robot with hard-real time considerations, i.e., the same as SRT incorporating critical battery level sensing, and also having the capacity to change the robot's normal operation to mission oriented, guaranteeing his survivability (if at least one charging zone was previously found).

4.1 Experimental Setup

As mentioned before, the experiments are performed using a modified version of YAKS. This simulation system has several different elements including: the robot simulator, neural networks, GA, and fuzzy logic based fitness.

Khepera Robot For these simulations, a Khepera robot was chosen. The robot configuration has two DC motors and eight (six front and two back) infrared proximity sensors used to detect nearby obstacles. These sensors provide 10 bit output values (with 5% random noise), which allow the robot to know in approximate form the distance to local obstacles. The YAKS simulator provides the readings for the robot sensors according to the robot position and the map (room) it is in. The simulator also has information for the different areas that the robot visits and the various obstacles (walls) or zones (home, charging zones) detected in the room. In order to navigate, the robot executes up to 1000 steps in each simulation, but not every step produces forward motion as some only rotate the robot. If the robot has no more energy, it freezes and the simulation stops.

Artificial Neural Network The original neural network (NN) used has eight input neurons connected to the infrared sensors, five neurons in the hidden layer and two output neurons directly connected to the motors that produce the robot movement. Additionally, in our real-time extensions we introduce another input neuron connected to the battery sensor (activated by SRT and HRT).

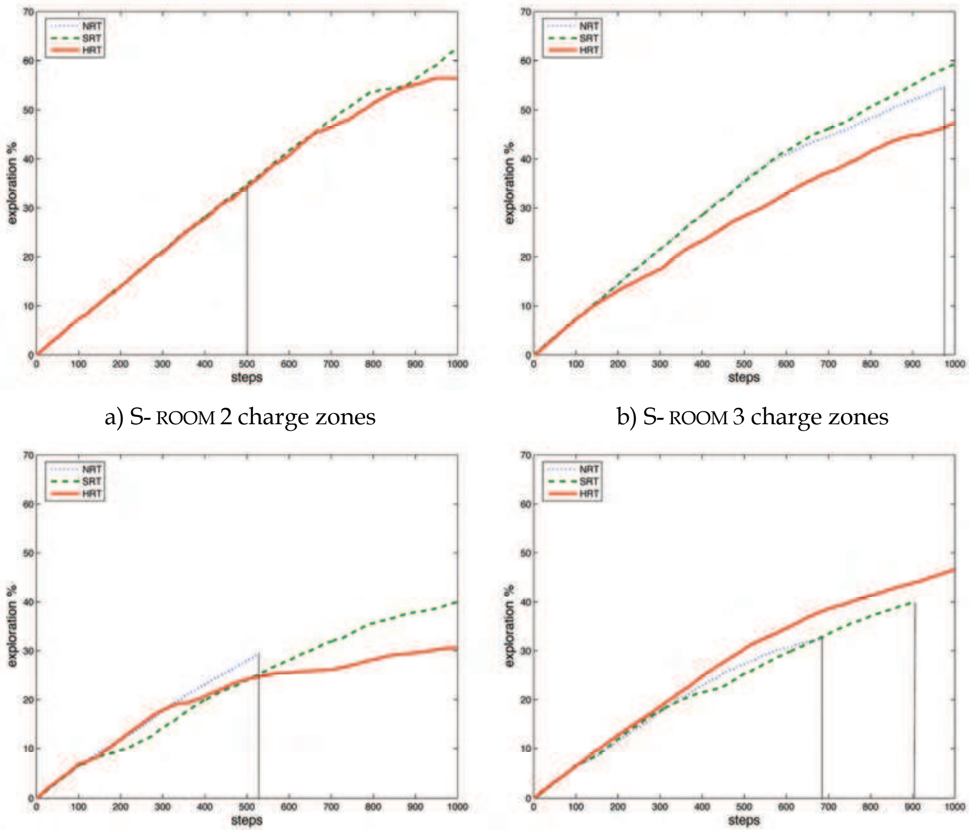
Genetic Algorithm A GA is used to find an optimal configuration of weights for the neural network. Each individual in the GA represents a NN which is evolving with the passing of different generations. The GA uses the following parameters:

- Population size: 200
- Crossover operator: random crossover
- Selection method: elite strategy selection
- Mutation rate: 1%
- Generations: 100

For each room (see Fig. 7) we trained a robot up to 400 steps, considering only configurations with 2 or 3 charging zones, i.e. shutting down zone 3 for 2-zones simulations. Startup battery level allows the robot to finish this training phase without recharging requirements.

Finally, we tested our algorithms in each room up to 1000 steps, using the previously trained NN for each respective room. The startup battery level was set to 80 (less than 50% of it's capacity), which was insufficient to realize the whole test without recharging.

4.2 Experimental Results



a) S- ROOM 2 charge zones b) S- ROOM 3 charge zones

c) H- ROOM 2 charge zones d) H- ROOM 3 charge zones

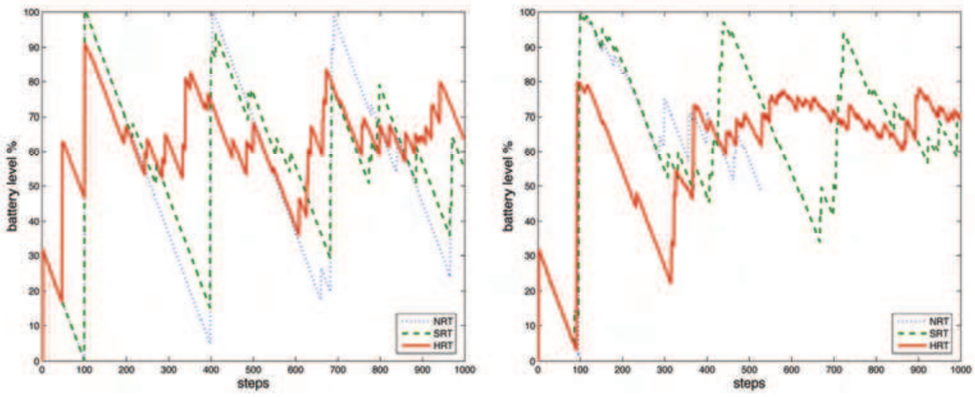
Figure 8. Exploration behaviour

We chose the S-ROOM and H-ROOM to show results for a simple and complex room respectively, which are representative behaviors of our approach.

In Fig. 8 we show the robot's exploration behavior for selected rooms. Each curve in the graph shows the average value of 10 executions of the same experiment (deviation between experiment iterations was very small, justifying only 10 executions). Let $surv(\alpha)_i$ the survivability of the experiment instance i of algorithm α, we define $surv(\alpha)$ as the survivability of an experiment applying algorithm α as the worst case survivability instance of an experiment, i.e.

$$surv(\alpha) = \min_{i=1,\ldots,10} [surv(\alpha)_i] \tag{1}$$

Please note that the end of each curve in Fig. 8 denotes the survivability of the respective algorithm (for better readability, we mark NRT survivability with a vertical line). Reaching step 1000 (the maximum duration of our experiments) means that the robot using the algorithm survives the navigation experiment. Finally, in Fig. 9 we show a representative robot's battery level. Monitoring was made during test phase in a H-ROOM with 3 charging zones.



a) S- ROOM 3 charge zones b) H- ROOM 2 charge zones

Figure 9. Battery Behaviour

4.3 Discussion

The results of our experiments are summarized below:

Survivability: As shown in Fig. 8, SRT and HRT algorithms give better reliability of completing missions than the NRT method, independently of the rooms (environments) we use for testing (see Fig. 7). As expected, if fewer charging zones are provided, NRT has a less reliable conduct. Please note that as shown in Fig. 9, NRT is also prone to battery depletion risk and does not survive in any case.

When varying the room complexity, i.e. 8(b) and 8(d), real-time considerations have significant impact. Using SRT, a purely behavior-based driven robot (with the additional neuron and motivation), improves it's performance. The SRT method does not guarantee Survivability since without changing the robot operation from behavior based to mission oriented the robot is prone to dying even with a greater number of recharge zones (as seen in 8(d)). Finally, we conclude that despite the uncertainty introduced by soft-computing methods, HRT (e.g. the hybrid algorithm), in general is the best and safest robot control method from a real-time point of view.

Exploration Environment: As can be seen in Fig. 8 safer behaviors means slower exploration rates (more conservative), up to 12% slower in our experiments. When comparing NRT with SRT, the exploration rates are almost equal in simple environments. In more complex rooms, SRT exploration is slower than NRT (due to battery observance). However, because of SRT having better survivability on the whole it's performance wins over NRT. If we compare NRT with HRT, exploration performance also favors NRT, wich could be explained given HRT conservative battery management (see Fig. 9).

Given 2 charge zones, HRT behaves differently in environments of varying complexity (up to 25%) which could be attributed to the complexity of the returning path to the nearest charging zone and loosing steps in further exploration. This phenomena becomes less notorious when increasing the number of charging zones (more options for recharge).

5. Conclusions and Future Work

In this work we investigate real-time adaptive extensions of our fuzzy logic based approach for providing biologically based motivations to be used in evolutionary mobile robot learning. We introduce active battery level sensors and recharge zones to improve robot's Survivability in environment exploration. In order to achieve this goal, we propose an improvement of our previously defined model (e.g. SRT), as well as a hybrid controller for a mobile robot (e.g. HRT), combining behavior-based and mission-oriented control mechanisms.

These methods are implemented and tested in action sequence based environment exploration tasks in a Khepera mobile robot simulator. Experimental results shows that the hybrid method is in general, the best/safest robot control method from a real-time point of view. Also, our preliminary results shows a significant improvement on robot's survivability by having minor changes in the robot's motivations and NN.

Currently we are implementing a real robot for environment exploration to validate our model moving from simulation to experimentation. We are also introducing dynamic motivations schedules toward robotic behavior enhancement. Improving the dependability of HRT, we want to extend this control algorithm to safety-critical domains.

6. References

T. Arredondo, W. Freund, C. Muñoz, N. Navarro, & F. Quirós. (2006). Fuzzy motivations for evolutionary behavior learning by a mobile robot. *Lecture Notes in Artificial Intelligence*, 4031:462–471.

Mohannad Al-Khatib & Jean J. Saade. (2003). An efficient data-driven fuzzy approach to the motion planning problem of a mobile robot. *Fuzzy Sets Syst.*, 134(1):65–82.

Ronald C. Arkin. (1998). Behavior-Based Robotics. MIT Press.

Humberto Martínez Barberá & Antonio Gómez Skarmeta. (2002). A framework for defining and learning fuzzy behaviors for autonomous mobile robots. *International Journal of Intelligent Systems*, 17(1):1–20.

W. Freund, T. Arredondo, C. Muñoz, N. Navarro, & F. Quirós. (2006). Realtime adaptive fuzzy motivations for evolutionary behavior learning by a mobile robot. *Lecture Notes in Artificial Intelligence*, 4293:101–111.

S. Goodrige, M. Kay, & R. Luo. (1997). Multi-layered fuzzy behavior fusion for reactive control of an autonomous mobile robot. In *Proceedings of the Sixth IEEE International Conference on Fuzzy System*, pages 573–578, July 1997.

Ahmed Gheith & Karsten Schwan. (1993). Chaosarc: kernel support for multiweight objects, invocations, and atomicity in real-time multiprocessor applications. *ACM Transactions on Computer Systems*, 11(1):33–72, February 1993.

Frank Hoffmann. (2000). Soft computing techniques for the design of mobile robot behaviors. *Inf. Sci.*, 122(2-4):241–258.

W. Huitt. (2001) Motivation to learn: An overview. Technical report, Educational Psychology Interactive, Valdosta State University.

Kiyotaka Izumi & Keigo Watanabe. (2000). Fuzzy behavior-based control trained by module learning to acquire the adaptive behaviors of mobile robots. *Math. Comput. Simul.*, 51(3-4):233–243.

J.-S. R. Jang, C.-T. Sun, & E. Mizutani. (1997). Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence. NJ: Prentice-Hall. ISBN: 0-13-261066-3.

Rahul Kumar Jha, Balvinder Singh, & Dilip Kumar Pratihar. (2005). On-line stable gait generation of a two-legged robot using a genetic-fuzzy system. *Robotics and Autonomous Systems*, 53(1):15–35, October 2005.

Kurt Konolige, Karen Meyers, and Alessandro Saffiotti. (1992). Flakey, an autonomous mobile robot. Technical report, Stanford Research Institute International, July 1992.

Hermann Kopetz. (1997). Real-Time Systems: Design Principles for Distributed Embedded Applications. Kluwer Academic Publishers, Norwell, MA, USA.

Makoto Kern & Peng-Yung Woo. (2005). Implementation of a hexapod mobile robot with a fuzzy controller. *Robotica*, 23(6):681–688.

Ji-Hong Li, Bong-Huan Jun, Pan-Mook Lee, & Seok-Won Hong. (2005). A hierarchical real-time control architecture for a semi-autonomous underwater vehicle. *Ocean Engineering*, 32(13):1631–1641, September 2005.

G. Motet & J.-C. Geffroy. (2003). Dependable computing: an overview. *Theor. Comput. Sci.*, 290(2):1115–1126, 2003.

Guido Maione & David Naso. (2003). A soft computing approach for task contracting in multi-agent manufacturing control. *Comput. Ind.*, 52(3):199–219.

H. Seraji & A. Howard. (2002). Behavior-based robot navigation on challenging terrain: A fuzzylogic approach. *IEEE Transactions on Robotics and Automation*, 18(3):308–321.

Homayoun Seraji & Ayanna Howard. (2002). Behavior-based robot navigation on challenging terrain: A fuzzy logic approach. *IEEE Transactions on Robotics and Automation*, 18(3):308–321, June 2002.

Bernhard Sick, Markus Keidl, Markus Ramsauer, and Stefan Seltzsam. (1998). A Comparison of Traditional and Soft-Computing Methods in a Real-Time Control Application. In *ICANN 98, Proc. of the 8th Int. Conference on Artificial Neural Networks*, pages 725–730, Sweden, September 1998.

Paul Tompkins, Anthony Stentz, & David Wettergreen. (2006). Mission-level path planning and re-planning for rover exploration. *Robotics and Autonomous Systems*, 54(2):174–183, February 2006.

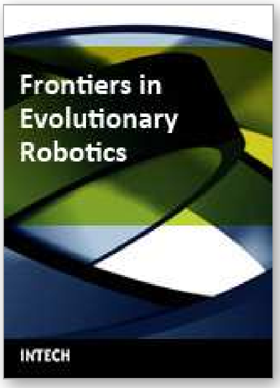
Jeen-Shing Wang & C.S. George Lee. (2003). Self-adaptive recurrent neurofuzzy control of an autonomous underwater vehicle. *IEEE Transactions on Robotics and Automation*, 19(2):283–295, April 2003.

YAKS simulator website: http://r2d2.ida.his.se/.

S. Yamada. (2005). Evolutionary behavior learning for action-based environment modeling by a mobile robot. Applied Soft Computing, 5(2):245–257, January 2005.

C. Zhou. (2002). Robot learning with ga-based fuzzy reinforcement learning agents. *Information Sciences*, 145(1), August 2002.

Tom Ziemke, Dan-Anders Jirenhed, & Germund Hesslow. (2005). Internal simulation of perception: a minimal neuro-robotic model. *Neurocomputing*, 68:85–104.



Frontiers in Evolutionary Robotics
Edited by Hitoshi Iba

ISBN 978-3-902613-19-6
Hard cover, 596 pages
Publisher I-Tech Education and Publishing
Published online 01, April, 2008
Published in print edition April, 2008

This book presented techniques and experimental results which have been pursued for the purpose of evolutionary robotics. Evolutionary robotics is a new method for the automatic creation of autonomous robots. When executing tasks by autonomous robots, we can make the robot learn what to do so as to complete the task from interactions with its environment, but not manually pre-program for all situations. Many researchers have been studying the techniques for evolutionary robotics by using Evolutionary Computation (EC), such as Genetic Algorithms (GA) or Genetic Programming (GP). Their goal is to clarify the applicability of the evolutionary approach to the real-robot learning, especially, in view of the adaptive robot behavior as well as the robustness to noisy and dynamic environments. For this purpose, authors in this book explain a variety of real robots in different fields. For instance, in a multi-robot system, several robots simultaneously work to achieve a common goal via interaction; their behaviors can only emerge as a result of evolution and interaction. How to learn such behaviors is a central issue of Distributed Artificial Intelligence (DAI), which has recently attracted much attention. This book addresses the issue in the context of a multi-robot system, in which multiple robots are evolved using EC to solve a cooperative task. Since directly using EC to generate a program of complex behaviors is often very difficult, a number of extensions to basic EC are proposed in this book so as to solve these control problems of the robot.