Behavioral Selection Using the Utility Function Method: A Case Study Involving a Simple Guard Robot

Mattias Wahde[1], Jimmy Pettersson[1], Hans Sandholt[1], and Krister Wolff[1,2]

[1] Department of Applied Mechanics, Chalmers University of Technology, 412 96 Göteborg, Sweden
{mattias.wahde, hans.sandholt, jimmy.pettersson}@chalmers.se
[2] Department of Microtechnology and Nanoscience, Chalmers University of Technology, 412 96 Göteborg, Sweden
krister.wolff@mc2.chalmers.se

Summary. In this paper, the performance of the utility function method for behavioral organization is investigated in the framework of a simple guard robot. In order to achieve the best possible results, it was found that high-order polynomials should be used for the utility functions, even though the use of such polynomials, involving many terms, increases the running time needed for the evolutionary algorithm to find good solutions.

1 Introduction

In behavior-based robotics (BBR) [1], the artificial brain of a robot is built in a bottom-up fashion, starting from simple low-level behaviors. An obstacle facing the behavior-based approach is the problem of behavioral selection, i.e. the problem of activating appropriate behaviors at all times. In simple robots, with small behavioral repertoires, the selection of behaviors (for activation) can be generated manually, which indeed is what is done in most methods for behavioral selection [4, 5, 6].

However, in robots with larger behavioral repertoires, specifying behavioral selection by hand is a daunting task, not least because of the difficulty in comparing, at all times and in all situations, the relative merits of several behaviors. Such comparison requries a common currency which, in economic theory and game theory, goes under the name *utility*, a concept that has also been introduced in ethology and, more recently, in robotics [3, 4].

In order to overcome the difficulties associated with behavioral selection, a method known as the *utility function* (UF) *method* has been developed [4]. In this method, behavioral selection is based on the value of utility functions that are *evolved* rather than hand-coded, thus minimizing the bias introduced by the user of the method.

In this paper, the UF method will be illustrated by means of an example, namely a simple simulated guard robot. In addition, the performance for various utility function specifications will be studied.

2 The Utility Function Method

Due to space limitations, only a brief description of the UF method will be given here. A more complete discussion of the method is available in [4]. In the UF method, each behavior $B_j, j = 1, \ldots, N$, where N is the number of behaviors, is associated with a utility function, whose variables are (a subset of) the state variables of the robot. The state variables are of three kinds: External variables, denoted s_i (e.g. the readings of IR sensors on the robot), internal physical variables, denoted p_i (e.g the readings of a battery sensor), and internal abstract variables, denoted x_i. The latter correspond to the readings of internal variables known as hormones in the UF method. In general, each utility function is given by a polynomial ansatz. For example, the ansatz for a second-degree polynomial utility function of two variables s_1 and x_1 is given by

$$U(s_1, x_1) = a_{00} + a_{10}s_1 + a_{01}x_1 + a_{20}s_1^2 + a_{11}s_1x_1 + a_{02}x_1^2. \tag{1}$$

The UF method is an arbitration method, i.e. a method in which one and only one behavior is active at any given time. Behavioral selection is simple in the UF method: at all times, the behavior whose utility function takes the highest value is activated. The problem, of course, is to specify the utility functions so as to generate purposeful and reliable behavioral selection. In the UF method, this is done using an evolutionary algorithm (EA). As in any EA, a fitness function must be specified. In the UF method, the fitness is often associated with the execution of a given task behavior, the other behaviors being considered as auxiliary behaviors, i.e. behaviors which are needed (such as battery charging), but which do not increase the fitness of the robot. Once the fitness function has been specified, the task of the EA is thus to set the coefficients of the N polynomial utility functions.

An interesting question in this regard concerns the number of such coefficients, which, in turn, determines the complexity of the problem that the EA must solve. In general, it can be shown that a polynomial function of n variables and of degree p contains

$$\binom{n+p}{p} = \binom{n+p}{n} \tag{2}$$

distinct terms. (For example, in Eq. (1), $n = 2$ and $p = 2$, so that the number of terms, according to Eq. (2), equals $\binom{4}{2} = 6$).

3 Case Study: A Simple Guard Robot

As a case study, consider a simple simulated guard robot whose task it is to patrol the arena shown in the left panel of Fig. 1. The arena contains numerous obstacles in the form of pillars, as well as three battery charging stations, located at corners in the arena. The simulated robot is a differentially steered, two-wheeled robot, with two DC motors. The robot will be equipped

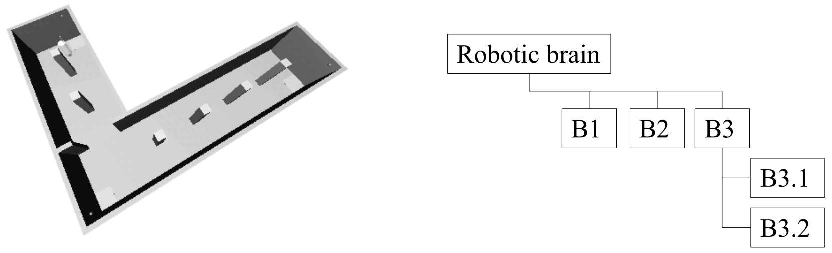

Fig. 1. Left panel: The arena patrolled by the guard robot. Right panel: Behavioral hierarchy of the robotic brain.

with five behaviors, namely *straight-line navigation* (B_1), *obstacle avoidance* (B_2), *energy maintenance* (B_3), *corner seeking* ($B_{3.1}$), and *battery charging* ($B_{3.2}$). In the UF method, as implemented in the UFLibrary software library currently under development at Chalmers University of Technology, behaviors are organized in hierarchies, as indicated in the right panel of Fig. 1. Utility functions are compared on a level-by-level basis. Thus, for each time step, it is determined which of the three functions U_1, U_2, and U_3 takes the highest value. If it happens to be U_3, the comparison of $U_{3.1}$ and $U_{3.2}$ will determine which of these two behaviors is active.

In B_1, the robot simply moves in a straight line, by setting its motor outputs to equal values. In B_2 the robot turns until no obstacle is visible in front of it, and then stops. If $B_{3.1}$ is active, the robot will rotate in an attempt to find a charging station (each of which is associated with an IR beacon detectable by a sensor on the robot). In $B_{3.2}$ the robot remains at a standstill, charging the batteries *if* it happens to be at a charging station.

Since the task of the robot is to cover as much of the arena as possible, a suitable fitness measure is simply the time spent in the navigation behavior B_1. In order to avoid rapid swapping between behaviors, a slightly modified fitness function was used, however, in which the robot only obtains a fitness increase if it spends at least one full second executing B_1.

Due to space limitations, the full ansatz for each utility function will not be given here. Suffice it to say that the number of variables in $U_1, U_2, U_3, U_{3.1}$, and $U_{3.2}$ were 4, 3, 3, 2, and 1, respectively. Thus, for degree p, the number of polynomial coefficients that must be determined by the EA equals

$$n_c = \binom{4+p}{4} + 2\binom{3+p}{3} + \binom{2+p}{2} + \binom{1+p}{1}. \tag{3}$$

4 Simulation program

A simulation program was written in Delphi object-oriented Pascal [2] using the UFLibrary software package. The UFLibrary provides a general implementation of the UF method, handling all issues involving the evolution of polynomial utility functions for behavioral selection. The task of the user is to provide (1) the constituent behaviors for the behavioral repertoire, (2) the polynomial degree p of the utility functions, (3) a fitness function, (4) the arena, in the form of a text file readable by the UFLibrary, (5) the definition of the robot body and the general structure of its brain (as shown in the right panel of Fig. 1), also in the form of a text file in a given format. The specification of the robotic brain also involves specifying the state variables included in each utility function polynomial.

5 Results

For simplicity, the simulations were performed without any sources of noise. Each evaluated individual was allowed a maximum simulation time of $100s$. However, the evaluation of a simulated robot was terminated directly in case of collisions with obstacles or if the on-board battery became fully discharged. In each run, 10,000 individuals were normally evaluated, even though some extended runs were carried out as well (see below).

Four different polynomial degrees were investigated, namely $p = 1, 2, 3$, and 4, for which the number of polynomial coefficients (n_c) equals 18, 44, 89, and 160, respectively. In order to allow a fair comparison between the performance of the EA for different values of p, mutation rates (p_{m}) were set to $1/n_c$ or $3/n_c$. The population sizes n_{p} were set to 20 or 100 individuals. In the UFLibrary, the crossover procedure swaps entire polynomials between chromosomes. Here, the crossover probability p_{c} was set to 0.2 or 0.8. Furthermore, tournament selection was used, with the tournament size n_{t} set to 10% of the population size. Thus, a total of $4 \times 2 \times 2 \times 2 = 32$ parameter combinations were investigated. Furthermore, because of the stochasticity of EAs, several (N_R) runs had to be performed for each setting, in order to form a reliable average. Since a typical run lasted for a few hours, N_R was limited to 5-7, resulting in a total of around 200 runs.

Due to the richer dynamical structure accessible in runs with large p, it could perhaps be expected that these runs would outperform those with lower p values. However, no such simple trend was found: The averages (over all runs with a given value of p) of the best fitness values found after 10,000 evaluated individuals, were found to be 10.36, 9.33, 10.31, and 9.43, for $p = 1, 2, 3$, and

Table 1. Averages of the best fitness values found after 10,000 individuals, for runs with different values of the polynomial degree p and the mutation rate p_m.

	Polynomial degree			
p_m	1	2	3	4
$1/n_c$	8.973	6.396	7.255	5.291
$3/n_c$	11.76	11.73	13.35	13.59

4, respectively. As can be seen in Table 1 there is, on the other hand, a strong trend in favor of the larger of the two mutation rates. A similar, albeit weaker, trend (not shown) was also found in favor of large population sizes, whereas no discernible trend was encountered for the crossover probability. Note also that, within the categories shown in Table 1, the spread between different runs was quite large, with the majority of runs reaching rather low fitness values.

6 Discussion and conclusion

While the average results obtained from the runs performed here show only very little variation with the polynomial degree p, this does not necessarily imply that the choice of p does not matter. Two possible interpretations of the results are that (1) perhaps larger values of p do indeed make it possible to achieve better results but that *finding* such solutions becomes progressively more difficult as p is increased (due to the large increase in the size of the search space), or (2) maybe $p = 1$ or 2 is sufficient for the problem at hand and that, in the runs with larger p, the coefficients in front of the third and fourth-order terms are simply eliminated by the EA. However, an inspection of those coefficients showed the latter *not* to be the case.

Table 2. Averages of the three best fitness values found for each polynomial degree p in the extended runs.

Polynomial degree			
1	2	3	4
23.24	37.23	47.91	49.33

Evidence in favor of the first interpretation can be found, on the other hand: As shown in Table 1, the increase in performance as the mutation rate is raised is stronger for large p values than for small ones, indicating that the larger p values require a more thorough inspection of the search space before the best solutions can be found. In order to test this tentative conclusion further, several extended runs were performed, the results of which are summarized in Table 2. Note that the extended runs differed somewhat in length: The number of evaluated individuals was on the order of 50,000 to 100,000. For this reason, the table shows an average of the three best results obtained for each p. In Table 2, the results clearly show an increase in performance as p is raised. Thus, it may be concluded that the choice of p strongly influences the quality of the results that can be achieved, but that the benefits of larger p values only become evident after the evaluation of a large number of individuals.

Finally, note that the chosen fitness measure (time spent in behavior B_1) does not lead to an incentive for the robot to explore the whole area. However, such solutions were indeed found by the EA in some runs. One example is shown in Fig. 2, with a fitness value equal to 44.57 found after the evaluation of 41,400 individuals.

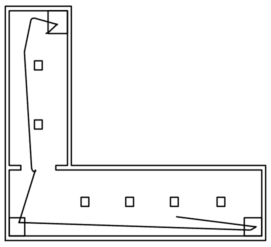

Fig. 2. The motion of one of the best robots found by the EA. The squares in the upper right, lower left, and lower right corners represent the charging stations.

Acknowledgments

The authors wish to thank the Carl Trygger foundation for financial support for this project.

References

1. Arkin, R.C., Behavior-based robotics, MIT Press, 1998
2. `http://www.borland.com/delphi`
3. McFarland, D. and Bösser, T. Intelligent behavior in animals and robots, MIT Press, 1993
4. Wahde, M., A method for behavioural organization for autonomous robots based on evolutionary optimization of utility functions, Journal of Systems and Control Engineering, 217, pp. 249–258, 2003
5. Maes, P., How to do the right thing, Journal of Connection Science, 1, No.3, pp. 291–323, 1989
6. Blumberg, B.M., Action selection in Hamsterdam: Lessons from ethology, In: From Animals to animats 3, Proc. of the 3rd Int. conf. on simulation of adaptive behavior (SAB94), MIT Press, 1994