Bootstrapping Aggregate Fitness Selection with Evolutionary Multi-Objective Optimization

Shlomo Israel and Amiram Moshaiov

Faculty of Engineering, Tel-Aviv University, Israel
shlomois@post.tau.ac.il, moshaiov@eng.tau.ac.il

Abstract. Aggregate fitness selection is known to suffer from the bootstrap problem, which is often viewed as the main inhibitor of the widespread application of aggregate fitness selection in evolutionary robotics. There remains a need to identify methods that overcome it, while requiring the minimum amount of a priori task knowledge from the designer.

We suggest a novel two-phase method. In the first phase, it exploits multi objective optimization to develop a population of controllers that exhibit several desirable behaviors. In the second phase, it applies aggregate selection using the previously obtained population as the seed. The method is assessed by two non-traditional comparison procedures. The proposed approach is demonstrated using simulated coevolution of two robotic soccer players. The multi objective phase is based on adaptation of the well-known NSGA-II algorithm for coevolution. The results demonstrate the potential advantage of the suggested two-phase approach over the conventional one.

1 Introduction

A major goal of Evolutionary Robotics (ER) is to develop methods for automatically synthesizing autonomous robot systems. However, the majority of ER research has used fitness functions which incorporate moderate to high levels a priori knowledge about the task [1]. Solving generally complex problems without the incorporation of a priori knowledge is still an open problem. Specifically, aggregate fitness, which bases selection only on success or failure to complete the task (a very low level of incorporated a priori knowledge [1]), is known to suffer from the bootstrap problem, in which randomly initialized populations have no detectable level of fitness and thus cannot be evolved [1]. Overcoming this problem is often considered as one of the main challenges of ER [2].

Our proposed approach is associated with two existing methods: (a) one that initially relies on a bootstrapping component and later gives way to aggregate selection [3], and (b) one that uses Multi Objective Optimization (MOO) for incremental evolution, relying on a pre-defined decomposition of the task into sub-tasks [4].

We suggest a novel method building upon these concepts. First, it exploits MOO to evolve a population of controllers which exhibit several useful, non-task specific, behaviors (explore environment, avoid obstacles, etc.). Secondly, it applies aggregate

C.A. Coello Coello et al. (Eds.): PPSN 2012, Part II, LNCS 7492, pp. 52–61, 2012.

selection using the previously obtained population as the seed population. This reduces considerably the probability of the population from being sub-minimally competent. As MOO supports diverse search, the overall proposed method remains non-tailored in nature, and thus should scale better to complex problems.

The contributions of the current work are:

1. Proposing and analyzing a novel method for dealing with the bootstrap problem in ER, which offers a more generic approach than common practices.
2. Expanding a recently suggested procedure (equal-effort comparison), and utilizing a unique one (end-game comparison), for comparing co-evolutionary methods.
3. Introducing a slightly modified version of the well established NSGA-II algorithm that makes it suitable for use in co-evolution.

To demonstrate the proposed method, this work employs a competitive coevolution task, namely a soccer game, for which aggregate selection is a natural choice.

The reminder of this paper is organized as follows: Section 2 provides some relevant background. Section 3 presents the comparison procedures and the proposed method. Section 4 provides details on the simulation study and the outcomes of comparisons. Finally, section 5 provides conclusions and future research suggestions.

2 Background

2.1 Bootstrap Problem

When trying to solve a complex task using a high-level reward function, practitioners of evolutionary algorithms, and other search and optimization methods, often encounter a situation where no initial search pressure exists. Particularly in ER studies, such a bootstrap problem is said to occur if all individual controllers in the initial population are scored with null fitness [5], or with no detectable level of fitness [1], prohibiting the onset of evolution. Overcoming this problem is one of the main challenges of ER [2]. Mouret & Doncieux describe the attempts to overcome the bootstrap problem as different schemes of incremental evolution [2]. Nelson et al. [1] and Mouret & Doncieux [2], independently, suggested categories of the available schemes. Here we refer to that of [2] including four categories: staged evolution, environmental complexification, fitness shaping and behavioral decomposition.

In [3] the bootstrap problem is dealt with by adding a bootstrapping component that is active only when the population is sub-minimally competent. This strategy fits well into the fitness shaping category. In [2] and [4] the bootstrap problem is addressed by adding different objectives to the original fitness function. This approach may be categorized as a fifth category that we hereby call multi-objectivization [6]. In [6] multi-objectivization denotes solving difficult single objective problems with local optima by elevating fitness dimensionality. Here we expand this notion to generally supporting solution of numerical difficulties by adding objectives.

2.2 Multi-objective Evolution

ER problems may involve trade-offs between different and possibly conflicting objectives. In such cases the designers may set the goals as part of their problem definition and obtain a Pareto-optimal set of controllers (e.g. [7]). Here the interest in contradicting objectives, and in the corresponding non-dominated set, is different. We explore their potential to serve as a booster for a single objective evolution. Existing Multi-Objective Evolutionary Algorithms (MOEAs) should support the initial phase of the suggested search method. For this purpose, we slightly modified the well established NSGA-II algorithm [8] to suit it for use in co-evolution (see section 3.1).

2.3 Khepera Robot Soccer

Aggregate selection is a natural choice for competitive co-evolution. Thus, we chose to experiment with a domain from this category. The experimental work reported herein is conducted on a robotic soccer-like game following [9]. In [9], two simulated Khepera robots are competing to scoring more goals than each other. In Fig. 1. the robots are depicted as pacman-like symbols chasing the light colored circle (ball).

The black sector within each robot's symbol shows the Field Of View (FOV) of its ball sensor. The inputs to the controller are sensor primitives including: proximity, ball direction and width (when in FOV), "stuck", and "goal direction". The outputs are motor primitives: translation speed, rotation speed, and rotation direction. The controller is a fixed-structured tree of behavior modules. The top behavior modules act as arbitrators, propagating control resolutions downwards, and the bottom modules activate primitive behaviors (motor primitives).

In [9], the fitness for one round was the sum of three components. The first component is meant to dominate the fitness if a goal is scored (as in aggregate selection). The second component favors earlier goals, and the last component favors the robots that stay close to the ball (a bootstrapping component). The authors indicate that the design of the function was a guess based on intuition and it "turned out to serve its purpose". This type of selection employs fitness shaping for bootstrapping, leaving us with an opportunity to change the method and examine if it is beneficial.

A practical factor in choosing this domain was that its source code has been available through a website provided in [9]. It made it possible for us to test alternative evolutionary schemes against the original one. We did encounter some problems with the simulator and introduced corrections, but nothing on the high-level was changed.

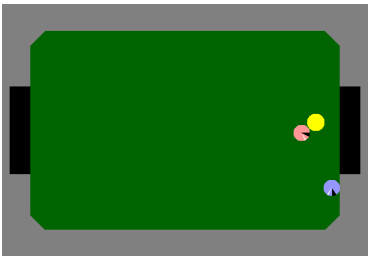

Fig. 1. A snapshot of the domain (as obtained using the website provided in [9])

3 Methodology

3.1 Comparison Procedures

In section 4 of this study several methods are compared, including random search vs. co-evolution, and the proposed two-phase approach vs. co-evolution. In both cases co-evolution is done as in [9]. To compare results the following techniques are used. Their implementation is available at: `http://bit.ly/LC0LYA`.

Equal-Effort Comparison. Several methods have been utilized to measure progress in coevolution, of which most popular are Masters Tournament [11] and Dominance Tournament [12]. But lately these methods have been criticized to measure historical progress while implying to measure global progress [13]. According to [13], the common fallacy of these methods is that they measure performance against previous opponents – that is, the sequence of successive opponents against which they have evolved. In this case, the training set is being used as a test set.

We decided to turn to alternative methods that do not involve a comparison of individuals with the opponents with which they have coevolved. The underlying assumption here is that separation of co-evolved populations, during a posteriori analysis, will bring us closer to assessing the global progress instead of the historical one.

In this context we utilize the equal-effort comparison technique [13]. This technique is designated to compare the performance of two methods by opposing the champions of one method (its hall of fame) against the champions of the other, and tracking the respective number of victories over the course of coevolution, using several independent runs. The competition is conducted between individuals that were obtained after an equal number of evaluations, thus the name equal-effort comparison.

On top of the basic graph that results from this technique we added some aspects of our own. These include the addition of 95% confidence intervals to the graph, and running a moving average for 5 generations to smoothen the curve. Also, we used a stop condition, which is introduced in the following sub-section, to decide how many independent runs are enough.

End-Game Comparison. Reaching statistical significance is a justifiable requirement; yet, it is not always customary in ER studies. With this respect, we introduce a technique for statistically comparing the performance of two methods by opposing their best performing representatives. The proposed technique follows the principal not to compare individuals with the opponents with which they have coevolved, similarly to the previous technique. As seen in Fig. 2. , for each method we gather the best 5 individuals out of the final population of each run, in a set, which is denoted S^i_j, where i is the method index (i=A or B) and j is the run index (j=1,…,N). We wish to ensure that N is a statistically significant number of runs but to keep it to a minimum. Thus, N is subject to a lower limit. An upper limit for N is also used to manage the computational efforts.

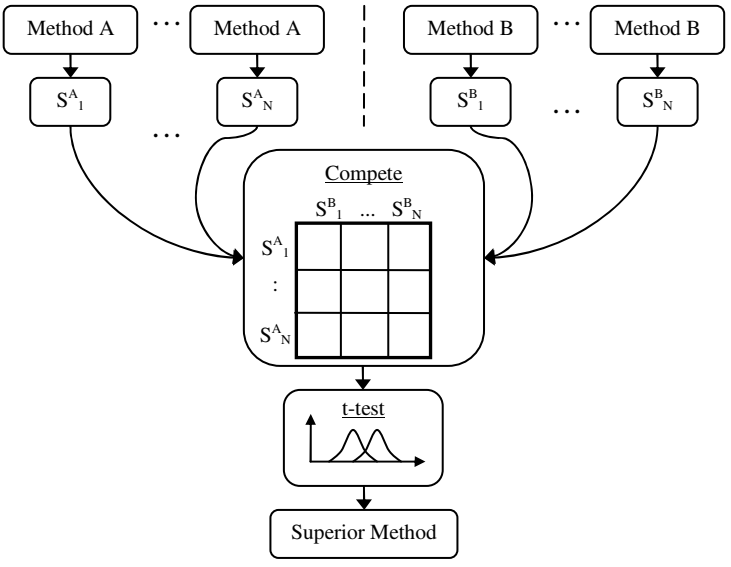

Fig. 2. The sequence of end-game comparison

At i=1, 5x5 competitions are carried out between all pairs from the two sets S^A_1 and S^B_1. At i=2 additional competitions are carried out to complete all possible pairings with the newly introduced individuals and with the existing ones, from i=1. The accumulated results are depicted as a matrix in the 'Compete' block in Fig. 2. . At each change of the run index we calculate an averaged fitness within each best-5 set. To reach statistical significance we run a t-test using the obtained averaged fitness values and cease if the p-value reaches 0.05. If the procedure has ceased before N reached its upper limit then the winning method is the one with the higher mean fitness. Otherwise, the comparison is declared as indecisive.

3.2 Two-Phase Method

Motivated by the idea that MOO supports diverse search, as demonstrated in [2] and [4], we propose a different use of MOO as a novel approach for bootstrapping. Some a priori knowledge is still required, as in any bootstrapping method. However, we aim to make our method more generic and less sensitive to disruptive assumptions.

Our approach is comprised of two phases. First, it exploits MOO to develop a population of controllers evolved to exhibit several desirable behaviors. Secondly, it applies aggregate selection using the previously obtained population as the seed population. We propose to choose the objectives for the first phase from a set of generally desirable behaviors in ER. These may be ‘explore environment’, ‘avoid obstacles’ and ‘approach target’. Of course these behavioral categories should be realized differently for specific domains, but we claim it is not difficult to design such objectives following a generic guideline - encourage useful behaviors which are not task specific. In Table 1, for instance, we suggested realizations for the domain used in this study.

In contrast to [2] and [4], we propose to multiobjectivize the problem only in early stages of evolution, disregarding the original objective in principal, and then proceed with the original objective alone. This approach is applicable in a wide variety of situations because it doesn't alter the whole algorithm but rather adds a preliminary phase.

For the MOO phase we used an open-source java-based framework, namely jMetal [10]. We slightly modified the well established NSGA-II algorithm to make it suitable for use in co-evolution. We call this variation ceNSGA-II, which means co-evolutionary NSGA-II. The difference lies within how the offspring are evaluated. We let the offspring play three matches (to reduce the influence of randomness): two against themselves and one against each other, such that their objective-values are determined according to the worst score of these three trials. A short examination suggested that in this manner the population is driven towards desirable behaviors.

4 Simulation Study

4.1 Random Search vs. Co-evolution

An initial confidence examination has been conducted to illustrate that co-evolution offers a clear advantage over random search. For this purpose two populations have been co-evolved for a fixed amount of generations, using the procedure of [9]. Then we generated a random population using the same amount of evaluations as was required for co-evolution. The evaluations of the randomly generated individuals served the purpose of establishing the inner ranking of the random population (about 5-10 evaluations per individual to extract the fittest out of a population of 10-20 thousand individuals).

Next, we ran the end-game comparison technique between co-evolution and random search. The comparison was done after 65,000 evaluations (equivalent to 50 co-evolution generations). The number of runs, N, was set in the range of [10, 30]. Meaning that the stop condition was not checked if there were less than 10 entries and that at most 30 runs were made.

As expected, the co-evolutionary method has proven to be superior (after 10 runs with a p-value reaching 10^{-6}). This unequivocal result is not surprising when considering the size of the search space, which makes it hard for randomly generated individuals to succeed.

4.2 Objective Combinations

For the main purpose of this study we explore which objective combinations may potentially lead to better results. We begin by listing, in Table 1, several broad categories of desirable behaviors in ER, including: (a) explore environment, (b) avoid obstacles, and (c) approach target. Then, also in Table 1, we list some possible realizations of these behaviors, as applied to the current particular domain. The realizations are formulated as objectives to be minimized. In the rest of this paper we denote formula (1) by "centerMove" or obj. 1, formula (2) by "proxiSensor" or obj. 2, etc..

Table 1. Summary of suggested behavior categories and corresponding realizations for our particular domain. The realization formulas are objectives to be minimized and are negative by convention; therefore some formulas are normalized by offest and factors.

Behavior Categories	Domain Realizations	Realization Formulas (Objectives)
Explore environment	Maximize linear speed	$-\sum_{t=1}^{T}\lvert\text{centerMove}(t)\rvert$ (1)
Avoid obstacles	Minimize proximity sensors;	$\sum_{t=1}^{T}\sum_{i}\text{proxiSensor}_i(t)$ (2)
	Minimize "stuck" sensor	$\sum_{t=1}^{T}\text{stuck}(t)-1$ (3)
Approach target	Minimize diversion from goal;	$\sum_{t=1}^{T}\lvert\text{direction}(t)\rvert-180\,/\,180$ (4)
	Maximize ball size in FOV;	$-\sum_{t=1}^{T}\text{ballSize}(t)\,/\,64$ (5)
	Minimize diversion from goal when ball in FOV	$\sum_{t=1}^{T}\frac{\text{ballSize}(t)}{64}\cdot\frac{\lvert\text{direction}(t)\rvert-180}{180}$ (6)

Next, we employed ceNSGA-II with several bi-objective combinations. The bi-objective approach is used to make the problem easier to handle and analyze, yet the method is not restricted to such an approach. The realization formulas rely on sensory data available to the controller and are mostly self explanatory, except of formula (6). The later formula is the result of an amalgamation of two objectives that seemed reasonable to combine.

The process of evaluating the bi-objective combinations was carried out by examining the non-dominated fronts in objective space and determining which combinations lead to well diversified fronts. We have found "proxiSensor" to be unsuccessful since the ball also activates the proximity sensors. We also found that the objectives "direction" and "ballSize" work better together than separately. This is because it is favorable to head towards the goal when the ball is in close sight.

In Fig. 3., typical performance distributions are shown for two different bi-objective combinations. We should note that the combination {obj. 1, obj. 6} had better diversification not only among the shown two cases but also when compared with other combinations; thus we proceeded with it.

The following search parameters are relevant for the study in this section and the following one: for selection we used the same binary selection operator as in NSGA-II (jMetal's 'BinaryTournament2'); for mutation we used a uniform mutation operator (jMetal's 'BitFlipMutation' operator for integer representation) with probability of 1/638; for crossover we used jMetal's 'SinglePointCrossover' operator with probability 0.9. The operator types and values were adopted from jMetal's examples.

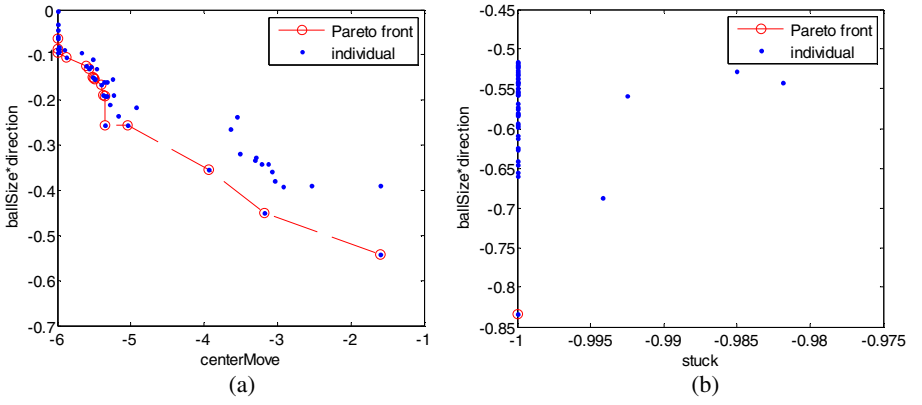

Fig. 3. Typical performance distribution for two bi-objective combinations: (a) {obj.1, obj.6 } and (b) {obj.3, obj.6 }. The instantaneous (local) Pareto front is marked with a dashed line and with circles around the performance vectors which comprise it.

4.3 Two-Phase vs. Co-evolution

Once reaching a seemingly promising objective combination, {obj. 1, obj. 6}, we set to examine its success in the full context of the two-phase method. First we utilized for this purpose the end-game comparison technique, using as competing methods the two-phase method and the original co-evolutionary method from [9], and setting N in the range [10,30].

Table 2 shows the results for the chosen bi-objective combination, for two independent comparisons which differ only in the total number of evaluations. The difference in fitness is due to the 5,000 evaluations used in the 1st phase. It is emphasized that the total number of evaluations in the co-evolution method is equal to the sum of evaluations made by the two-phase method, in both phases.

Table 2. End-game comparisons between the co-evolution and the two-phase methods. Two comparisons appear in the table separated by a dashed line. The 1st row, in each comparison, refers to the co-evolutionary method, and the 2nd to the two-phase method. The 3rd till the 6th columns show how many generations and evaluations were used in the MOO phase and the co-evolution phase respectively. The last two columns under 'comparison' show the average fitness and its difference as obtained in each method. The symbol * that appears in the last column indicates significance at the 5% level; the corresponding t-statistic appear in parentheses.

		MOO phase		co-evo. phase		comparison	
Comp. №	Method	gen.	eval.	gen.	eval.	Avg. fitness	Fitness diff.
1	co-evo.	0	0	50	65,000	1594.0	1247.8*
	two-phase	50	5,000	46	60,000	2841.8	(2.27)
2	co-evo.	0	0	200	250,000	2438.5	143.8
	two-phase	50	5,000	196	245,000	2582.3	(0.45)

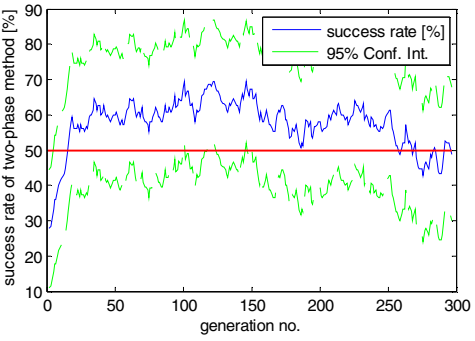

Fig. 4. Equal-effort comparison results showing the success rate of the two phase method and the corresponding 95% confidence intervals versus generations. Generations are counted in terms of co-evolution (not to confuse with the notion of generations in the MOO phase). Success in a match is counted when a champion of the two-phase method defeats the corresponding champion of the co-evolutionary method; disregarding the fitness difference that led to the victory. Here the success rate was calculated based on N=30 independent runs.

It is evident from the table that initially (after 65,000 evaluations – equivalent to 50 co-evolution generations) the two-phase method is superior. However, there is no such clear conviction in the longer case (after 250,000 evaluations – equivalent to 200 generations). To get an idea of the required computational effort we note that one generation is evaluated at approximately 7 seconds on a Core i5 CPU @ 2.66GHz.

An equal-effort comparison is carried out to better understand when the advantage of the two-phase method fades. The results for a 375,000 evaluations long run (equivalent to 300 co-evolution generations) are presented in Fig. 4.

As expected when there is no clear winner, the number of runs was 30. From the graph it is evident that the two-phase method rises to about 70% success rate in 100 generations, but then in about 80 generations it regresses to the 50% equilibrium. This analysis is consistent with the results present in Table 2. The results indicate that the two-phase method has an advantage over the original method in the early stages of co-evolution, but this advantage fades as co-evolution progresses.

5 Conclusions and Future Research

A novel bootstrapping approach is proposed based on multi-objectivization and a two-phase evolution scheme. The suggested approach is compared with a conventional one using co-evolution of robot controllers for a soccer game. The conventional method employs an intuitive domain-specific bootstrapping component, whereas the proposed approach aims to be generic.

In this study we put emphasis on carefully designed and statistically significant comparisons. Using such comparisons, our results confirm that the two-phase method is better in bootstrapping the evolutionary process. We also demonstrated that it is possible to encourage the acquisition of useful behaviors without losing valuable

portions of the search space. However, from the erosion in the advantage of the two-phase method we conclude that, at least in our simulation study, it did not succeed in expressing unique behaviors unreachable by conventional means after enough generations.

In regard to ceNSGA-II, it is recommended that in future research various alternatives to evaluate and compare offspring performances will be examined, and the co-evolution algorithm will be optimized. Regarding the MOO phase, it is conceivable that in more complex domains its positive effect will last longer and even persist. Further research is needed, with various domains and fitness aggregation, to further assess the proposed technique, based also on CPU time comparisons. Such studies are also expected to enable automatic selection of satisficing objective combinations in a generic manner.

References

1. Nelson, A.L., Barlow, G.J., Doitsidis, L.: Fitness functions in evolutionary robotics: A survey and analysis. Robotics and Autonomous Systems 57, 345–370 (2009)
2. Mouret, J.B., Doncieux, S.: Overcoming the bootstrap problem in evolutionary robotics using behavioral diversity. In: 2009 IEEE Congress on Evolutionary Computation, CEC 2009, pp. 1161–1168 (2009)
3. Nelson, A.L., Grant, E.: Using direct competition to select for competent controllers in evolutionary robotics. Robotics and Autonomous Systems 54, 840–857 (2006)
4. Mouret, J.B., Doncieux, S.: Incremental evolution of animats' behaviors as a multi-objective optimization. From Animals to Animats 10, 210–219 (2008)
5. Nolfi, S.: Evolutionary robotics: Exploiting the full power of self-organization. Connect. Sci. 10, 167–184 (1998)
6. Knowles, J.D., Watson, R.A., Corne, D.W.: Reducing Local Optima in Single-Objective Problems by Multi-objectivization. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) EMO 2001. LNCS, vol. 1993, pp. 269–283. Springer, Heidelberg (2001)
7. Moshaiov, A., Ashram, A.: Multi-objective evolution of robot neuro-controllers. In: 2009 IEEE Congress on Evolutionary Computation, CEC 2009, pp. 1093–1100 (2009)
8. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6, 182–197 (2002)
9. Østergaard, E.H., Hautop Lund, H.: Co-evolving Complex Robot Behavior. In: Tyrrell, A.M., Haddow, P.C., Torresen, J. (eds.) ICES 2003. LNCS, vol. 2606, pp. 308–319. Springer, Heidelberg (2003)
10. Durillo, J.J., Nebro, A.J., Alba, E.: The jmetal framework for multi-objective optimization: Design and architecture. In: 2010 IEEE Congress on Evolutionary Computation, CEC 2010, pp. 1–8 (2010)
11. Nolfi, S., Floreano, D.: Coevolving Predator and Prey Robots: Do "Arms Races" Arise in Artificial Evolution? Artif. Life 4, 311–335 (1998)
12. Stanley, K.O., Miikkulainen, R.: The dominance tournament method of monitoring progress in coevolution. In: GECCO 2002, pp. 242–248 (2002)
13. Miconi, T.: Why Coevolution Doesn't "Work": Superiority and Progress in Coevolution. In: Vanneschi, L., Gustafson, S., Moraglio, A., De Falco, I., Ebner, M. (eds.) EuroGP 2009. LNCS, vol. 5481, pp. 49–60. Springer, Heidelberg (2009)