Contents

1 Introduction

Nature has always been a source of inspiration in the field of robot design. Evolution has produced organisms that are perfectly adapted to their environments. Consider for example the emperor penguin: living in Antarctica, it faces one of the harshest environments on the planet, with temperatures down to -40°C and intense wind speeds. However, with its specifically adapted morphological features, like its layers of fat and feathers, the emperor penguin has no problem handling this rough environment. Its oily coat and streamlined body also makes it an expert swimmer, and the characteristic dark back and white belly provides good camouflage when hunting. [35] These traits were formed through millions of years of evolution, and leave the penguin well adapted to solve the problems in its environment. Since the results of evolution in nature is this specialized, simply copying it into a specific problem solving robot will probably lead to a design that includes traits unnecessary or even obstructive for solving the problem at hand. Evolutionary Robotics (ER) attempts to mimic the process of evolution instead of its biological results, to create robots that are evolved to handle the environment of a certain problem. Often the control parameters of an existing robot are improved using this method, but one of the final goals of ER is the evolution of both control parameters and morphology [32, 4]. Although this task is extremely difficult due to the vast number of iterations and the many parameters that are often needed to describe solutions and environments, one advantage of ER is that it is not restrained by biological processes, and can therefore include problem-specific algorithms such as lifetime learning in the evolutionary search. This is what constitutes the essential ideas behind Memetic Algorithms.

2 Memetic Algorithms

Evolutionary Algorithms (EAs) are population based optimization algorithms inspired by the basic concepts of biological evolution and genetics, such as natural selection and inheritance. Memetic Algorithms (MAs) use this as a basis, and in addition includes ideas about lifetime learning, such as Lamarcks theory of evolution and the Baldwin effect [1].

2.1 Cultural evolution

The word “meme” was introduced by Richard Dawkins in [7], and is used to describe a unit of human cultural evolution, equivalent to “gene” in biological evolution. A meme can be described as a small building block in some kind of knowledge or skill, that an individual can acquire by learning. This means that the meme can be modified during the lifetime of the individual, and then, if it is good/interesting enough, be passed on to the next generation. This separates memes from genes, as the latter will be transmitted to the next generation unaltered, if the individual’s fitness is good enough. With this concept of memes in mind, Moscato [20] defined the term Memetic Algorithms. MAs combine the analogies of genes and memes by including local search as a sort of life-time learning in a Genetic Algorithms (GA).

2.2 Genetic Algorithms

GAs are a subset of Evolutionary Algorithms, which again is a subset of Evolutionary computation (EC).

In general, EAs/GAs perform iterative search or optimazation of a problem by evolving a population of candidate solutions, or individuals, towards an optimum, by modifying these using mutation and/or crossover operations. The new individuals go through a selection process, where individuals with higher fitness values have a higher chance of being selected to the new generation. The same process is then repeated on the new generation, until a termination criterion is reached, i.e. a certain number of iterations have been performed. For a more detailed description of EAs and its variants, see [9].

Evolutionary computing, as part of the field of artifitial intelligence, is becoming an established research area in computer science, and includes a large set of techniques, including EAs and GAs. It has reached popularity due to its ability to perform fast global search on high complexity problems that require satisfactory, but not necessarily optimal, performance, and because it can be applied to a large number of problems without much altering. This can be done because of the fact that it makes few assumptions about the final solution, which also makes it relatively straight forward to implement on different systems.

2.3 Life-time learning

A well known disadvantage of GAs, and EAs in general, is that there is no guarantee that the global optimum will be found, as the whole process is characterized by stochasticity and lacks the ability to exploit local information. They do, however, guarantee a near optimum solution, as EAs perform well on global search due to their ability to explore large areas of the fitness landscape. The idea behind memetic algorithmss is to combine the explorative qualities of GA with the exploitative qualities of heuristic local search, so a global optimum should have a higher probability of being discovered with this hybrid global-local search approach. The application of local search is sometimes referred to as life-time learning, see [21]. There are two main models of life-time learning, Lamarckian and Baldwinian [16, 34]. Lamarckian learning transmits the improvement caused by local search back to the population by encoding the improved phenotype back to a genotype, which is then used in selection and reproduction in the normal GA way. Although Lamarcks theory of evolution has been more or less discarded as a correct description of biological evolution, computational evolution is not bound by biological constraints, and with an applicable phenotype to genotype encoding there is no reason why this could not be implemented. Baldwinian learning is based on the more biologically accepted mechanism of the Baldwin effect [1]. In this method, the fitness of the individual is altered after the local search and thereby affects the selection process, but the results of the local search are not inherited by the new generation. An early simulation of Baldwinian learning can be found in [12], where it was stated that learning alters the search space, and that the shape of the search space indicates if learning will be a positive influence or not. Most successful MAs to date implement Lamarckian learning [16]. Although it poses the additional challenge of encoding the phenotype into a corresponding genotype, the fact that the results of the local improvements are placed back in the population seems to give better results in general.

2.4 Overview of the process

Figure 1 on the following page shows an outline of a standard run of a memetic algorithm. The first step is to initialize a population. This is often done randomly to avoid bias, but can also be done using known information about the problem. Local search is then performed on the initial population, before the main loop is entered, starting with parent selection based on the current fitness values of each individual. In some configurations, not only fitness affects the parent selection, but also other parameters, like age. Mutation and/or recombination of the parents is then performed, generating new individuals which form the new generation, on which local search then can be done. All individuals now represent local optima, or near local optima depending on the configurations of the local search algorithm. If the implementation uses Lamarckian learning, the new individuals receives a new genotype encoded from their improved phenotype before continuing, otherwise only the fitness stays improved. A new population is then made through survivor selection of the old and the new individuals, which can be done using either an elitist or a generational approach. With an elitist algorithm, which is the more common, individuals of high fitness are kept in the new population, as apposed to a generational algorithm, where all individuals of the last population are replaced by the new. This is repeated until a specified termination criterion is reached, often after a certain number of iterations. For a more detailed explanation, see [9] or [22].

2.5 Local search and fitness landscapes

The success of the MA largely depends on the choice of local search. This is rarely a non-trivial choice to make, as the effectiveness of different local search algorithms varies over the local structures of the search space, in correspondance with the No Free Lunch Theorem [36]. Much research has been done on the topic of fitness landscapes and MAs, including [19, 16, 5, 26]. [19] emphasizes the importance of fitness landscape analysis when considering MAs, and suggests a few methods for determining both global and local structures. To find the best performing local search, random walk correlation analysis is used to analyze the local structure. A high correlation between neighbouring points indicates a smooth fitness landscape, whereas a low correlation suggests a rugged fitness landscape with many local optima. Correlations between local

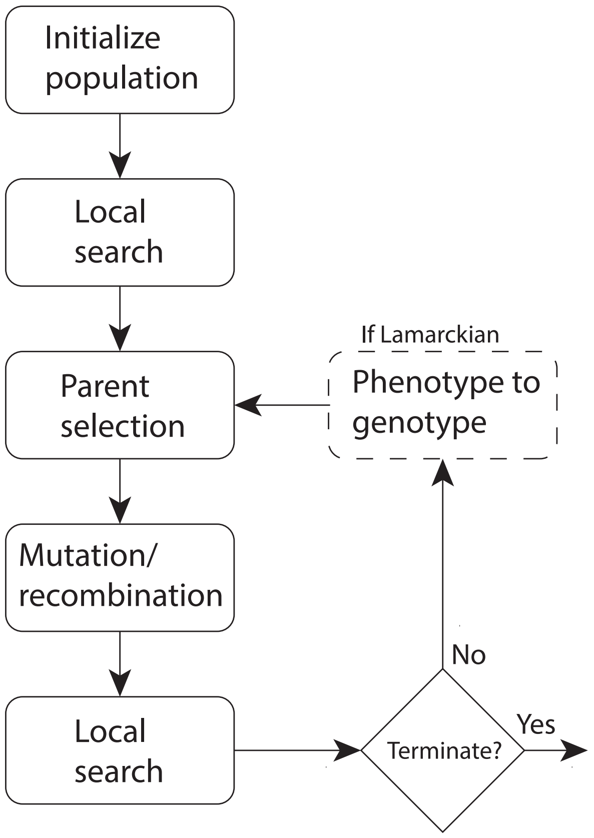

Figure 1: Overview of process

and global optimums are also examined, to determine the effectiveness of mutation versus recombination based MAs, and to denote the performance of MAs over certain types of landscapes compared to other heuristics. [5] uses the term *phenotypic plasticity* to describe the change in an inidividuals fitness due to lifetime learning, and demonstrates how this can have a smoothing effect on the fitness landscape. Their findings suggest that this effect is beneficial in rugged multipeaked landscapes, but that it may slow evolution down if the landscape is shaped by a simple function without multiple local optima. Random walk provides a baseline for their experiments, and is used as a way of measuring evolution rate on different fitness landscapes. [16] proposes benchmark analyses for connectivity structures for Lamarckian memetic algorithms, and attemps by this to introduce generality in this field, as most earlier research has been concerned with spesific problems and proof-of-concepts. Their results are similar to those in [19], confirming that connectivity/correlation and local optimum structure in general influences the performance of MAs notably. More specifically, local structures of the fitness landscape influence the effectiveness of the local search, while global structures affect the evolutionary meta-search. They suggest using a statistical analysis method, like random walk correlation analysis, to create an idea of the fitness landscape and thereby find the best local search algorithm. [26] explores the effects of shifting the balance between individual and population adaptation on changing environments, as a possible solution to limited computational power. This is done by introducing a "life-time parameter" which sets the degree of individual level versus population level adaptation, or exploitation versus exploration, respectively.

2.6 Adaptive MAs

As the choice of local search is of real importance, finding good methods for this has been some of the main topics in recent studies on MAs. [25] proposes a classification of adaptive MAs, and shows how adaptive MAs are capable performing more robustly than traditional MAs. This classification is revisited in [21], and an updated version is proposed. Four main categories are described; Adaptive Hyper-heuristic, where local search algorithms are coordinated using fixed rules; Meta-Lamarckian learning, where the success of the different local searches affects how often they are applied; Self-Adaptive and Co-Evolutionary, where local searches are evolved alongside the candidate solutions; Fitness Diversity-Adaptive, where fitness diversity is used to select the most appropriate local search algortihm. In [24], the term Meta-Lamarckian learning is used to describe methods that use multiple local search algorithms during a run of a Lamarckian MA. Their proposed approach uses a pool of local searches which compete and cooperate during the evolutionary meta-search, thereby avoiding having to manually select the best local search algorithm beforehand. On problems with *a priori* unknown fitness landscapes, this approach could be useful.

3 Evolutionary Robotics

Evolutionary robotics (ER) has yet to become an established part of mainstream robotics. To this day, robot design in general involves manual design of the robot's shape, and the use of machine learning for control policy optimization on the hand-designed robot. This process is extremely time consuming and demands a lot of resources, however, the results still outperform the current results obtained through ER [4]. Still, ER keeps gaining popularity for a number of reasons, one being that improving control policies and morphology can be done automatically while making few assumptions about the final system. This is mainly based on the formulation of a fitness function or a novelty search. Another advantage is that evolutionary algorithms can find solutions that are non-intuitive for human designers. This makes robotic design in ER more efficient, automatic design of robot control and/or morphology can and has created vast numbers of robots, obviously of varying quality. ER has also been shown to exploit rather than fight against morphological or environmental features, e.g. in [10], evolution of locomotion patterns on a monkey-like robot produced a pattern which exploited the momentum of the robots body, similar to what primates do when swinging from one tree to the next.

Another benefit of ER is that it can be interesting not only to roboticists, but also to biologists, as robots might evolve traits that are also seen in nature, and through this possibly help explain why these traits exist.

One disadvantage of ER, and EAs in general, is that the optimal solution is rarely found, and the number of iterations performed before a near optimal solution is found is often vast. Incorporating MAs in ER could lead to an improvement in this area, when looking at some of the promising research done on MAs.

See [4] for a more detailed overview of the field.

3.1 Evolving robots

So far, ER researchers have mainly been evolving control policies, often in the form of a neural networks. However, full automation of robot design is desirable, and some research has also been done on the evolution of morphology, or more ideally, morphology and control policy simultaneously, starting with [32, 31]. Sims used genetic algorithms to evolve both morphology and control of virtual creatures, which in the end were capable of swimming, walking, jumping or following a light. Similar simulated creatures were brought into the physical world through the GOLEM Project [27]. Although these early attempts at full automation were fairly simple, they show the potential to automatically create natural robots. A more recent experiment conducted by Cheney et al. [6] addresses the issue of lacking improvement since Sims [32], and present a new approach which includes multiple material types, like soft muscles and tissue. They use a CPPN-NEAT encoding [6] which is shown to produce more advanced creatures in simulation, and the multiple materials make evolution of more natural looking creatures possible.

3.2 Variants

In addition to classic legged or wheeled locomotive robots, there exists a wide array of variants suitable for artificial evolution, such as modular, swarm, and soft robotics.

3.2.1 Modular robotics

Modular robotics makes repetition or reuse of evolved sections or modules possible, which is also often seen in biological organisms, see [2, 30, 18]. Modularity seems a great advantage when evolving complex morphology, and could potentially increase the evolvability of robotic systems [2].

3.2.2 Swarm robotics

Swarm robotics takes its inspiration from social insects like ants, in other words, simple individuals that work collectively in a group. Early work includes [28], which focuses on evolving controllers for cooperative behavoir in small, homogeneous robots. The results showed that the robots developed distinct roles in the team, and worked together to complete a coordinated movement task.

3.2.3 Soft robotics

Soft robots are, as the term suggests, built using soft materials in addition to rigid parts. Such robots could potentially handle environments that standard discrete robots struggle with, i.e. by climbing walls or squeezing through holes. However, this flexibility also makes controller design difficult due to the fact that deformation of one part of the robot will consequently cause deformation of another part. Artificial evolution would then be ideal for controller or morphological design for robots of this kind. This is attempted in [29], where NVidia's PhysX is used to simulate evolution of soft robot gaits, as well as for soft-body modeling.

3.3 Challenges

There are still a number of unresolved challenges within ER which can explain why it has yet to produce a robot which is superior to one of manual design. Current research mainly focuses on these challenges.

3.3.1 Reality gap

The large number of iterations needed for artifitial evolution to produce good results is the reason why simulation is such an important part of the process. Ideally, evolution would be performed on the physical robotic system, but for most applications this process is too time consuming, and also involes a lot of wear and tear on the physical robot. The continuous and noise filled properties of the real world make creating a realistic physics based simulator a difficult task, and an inaccurate simulator would lead to badly transferable solutions due to the fact that evolution exploits all aspects of the environment. Solutions may become overly adapted to a simulated environment that does not accurately match the real world. This difference in performance between a simulated solution/controller and the transferred real world solution/robot is called "the reality gap". [13] found that adding noise to the simulation can create better correspondence between real world applications and simulations if the level of noise is appropriate, as noise blurs the fitness landscape in simulation. The more accurate a simulator is, the more time the optimization process will take.

In [37], a different approach is used, namely a back-to-reality algorithm, which does coevolution of simulator and robot/controller. This involves regular validations of the simulation model in the real world, followed by updates in fitness. [15] also uses a robot-in-the-loop approach, namely the transferability approach, but here a Pareto-based multiobjective evolutionary algorithm is used, where the objectives to be optimized are fitness and transferability. Transferability is measured using a simulation-to-reality disparity measure, which for most potential solutions is approximated using interpolation of a few real world measures. This approach ensures that the optimal solutions found in simulation are transferable, but it does not necessarily find the best real world solutions, which is yet to be done.

3.3.2 Nature of EAs

To achieve evolution of adequate controllers/robots in simulation the task environment must be well described, often demanding a large number of parameters. An increasing expressiveness rapidly increases the evaluation time of each robot, creating a complexity problem, and demanding extensive computational power. [4] mentions coevolution of robots and task environments and evolvability of algorithms as possible solutions to this problem.

Another key element in EAs is the fitness function, which describes the quality of a solution. It is near impossible to design an unbiased fitness function, resulting in biased solutions, even though one of the goals of EAs is to include as few assumptions as possible about the final result. One possible adaption of this can be to omit the fitness function and instead do novelty search [17]. Novelty search does not evalutate each individual on a certain performance like the fitness function, it rather compares the difference in functionality between new individuals and what has been discovered before. Significant difference is rewarded, leading to higher complexity, similar to natural evolution.

3.4 Resent work in ER

In addition to some of the work already mentioned, extensive research has been performed on the subject of ER. [8] lists two methods in ER as current trends, evolutionary aided design and online evolutionary adaptation.

3.4.1 Evolutionary aided design

Evolutionary aided design aims at using EAs to find promising strategies, and then using more traditional design techniques based on the results. In this way an evolutionary algorithm is used more as an analysis tool than for optimization, which can be useful for systems where possible optimization parameters are not obvious, possibly because of stochasticity or non-linearity. In [11], this technique is used to find controllers for a homogeneous swarm of micro air vehicles (MAVs). They first applied an evolutionary algorithm to automatically evolve neural controllers, then analysed the resulting behaviours and reverse-engineered these using hand-design to provide simple controllers that were easy to parameterize for different scenarios.

3.4.2 Online evolutionary adaptation

Online evolutionary adaptation involves continuously running an alogrithm on the robot, in order to online deal with possible changes on the robot or in the environment. This means that human intervention is unnecessary, which is obviously advantageous where such is unavailable, like in hazardous environments. [33] applied a form of online evolution to a population of robots, called embodied evolution, in which an evolutionary algorithm was distributed among a group of robots which then evolved through mutation and recombination between robots. An advantage of this approach is that evolution could be done outside simulation, thus avoiding transferability issues. Another is that evolution could be done "in the field" without human intervention. [3] also uses an online approach, by letting a legged robot detect changes in its own morphology, and then adapting the controller by synthesizing new models. In any case, the idea is that optimization is done whithout interfering with the robot performing its task. Since online evolution is a real-time operation, including MA could be advantageous as it has been shown that MAs can provide solutions more efficiently than plain genetic algorithms in some cases.

4 Combining Memetic Algorithms and Evolutionary Robotics

The fields of Memetic Algorithms and Evolutionary Robotics both lack maturity, so it is not surprising that very little research has been done on combining the two. Still, some work has been done on the field, including [23] and [14]. [23] applies a compact memetic algorithm to a cartesian robot for optimization of the control system, with good results. Their Memetic compact Differential Evolution (McDE) algorithm is designed for problems where high power computational components are unavailable, i.e. due to space/cost requirements or limited hardware, making it ideal for robotics. In [14], a hybrid genetic algorithm including bacterial foraging was used to optimize the parameters of the PID controller of an automatic voltage regulator. This algorithm showed promising results, and could pontentially be used for other similar optimization problems, such as the development of robot controllers.

Another reason for why this is a little explored field, could be that finding a good local search algorithm for ER is not trivial. The performance of MAs is as mentioned highly dependent on fitness landscape, which is usually unknown in ER. However, the stochastic local search used in [23] produced good results, which seems promising for other ER applications. Moreover, although the exact shape of the fitness landscape is mostly unknown, it will in most situations in ER be multi-peaked. MAs have demonstrated good performance on such landscapes, so further research on this is of interest. A possible solution to the problem of choice of local search could be to use an adaptive memetic algorithm, as resent research on this topic indicate that this could lead to good results on unknown fitness landscapes.

References

[1] J Mark Baldwin. A new factor in evolution. *American naturalist*, pages 536–553, 1896.

[2] Josh Bongard. Evolving modular genetic regulatory networks. In *Computational Intelligence, Proceedings of the World on Congress on*, volume 2, pages 1872–1877. IEEE, 2002.

[3] Josh Bongard, Victor Zykov, and Hod Lipson. Resilient machines through continuous self-modeling. *Science*, 314(5802):1118–1121, 2006.

[4] Josh C Bongard. Evolutionary robotics. *Communications of the ACM*, 56(8):74–83, 2013.

[5] E Borenstein, I Meilijson, and E Ruppin. The effect of phenotypic plasticity on evolution in multipeaked fitness landscapes. *Journal of evolutionary biology*, 19(5):1555–1570, 2006.

[6] Nick Cheney, Robert MacCurdy, Jeff Clune, and Hod Lipson. Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. In *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference*, pages 167–174. ACM, 2013.

[7] Richard Dawkins. The selfish gene. 1976.

[8] Stéphane Doncieux, Jean-Baptiste Mouret, Nicolas Bredeche, and Vincent Padois. Evolutionary robotics: Exploring new horizons. In *New Horizons in Evolutionary Robotics*, pages 3–25. Springer, 2011.

[9] Agoston E Eiben and James E Smith. *Introduction to evolutionary computing*. Springer, 2003.

[10] Dominic R Frutiger, Josh C Bongard, and Fumiya Iida. Iterative product engineering: Evolutionary robot design. In *Proceedings of the fifth international conference on climbing and walking robots*, pages 619–629. Professional Engineering Publishing, 2002.

[11] Sabine Hauert, J-C Zufferey, and Dario Floreano. Reverse-engineering of artificially evolved controllers for swarms of robots. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 55–61. IEEE, 2009.

[12] Geoffrey E Hinton and Steven J Nowlan. How learning can guide evolution. *Complex systems*, 1(3):495–502, 1987.

[13] Nick Jakobi, Phil Husbands, and Inman Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *Advances in artificial life*, pages 704–720. Springer, 1995.

[14] Dong-Hwa Kim and Ajith Abraham. A hybrid genetic algorithm and bacterial foraging approach for global optimization and robust tuning of pid controller with disturbance rejection. In *Hybrid Evolutionary Algorithms*, pages 171–199. Springer, 2007.

[15] Sylvain Koos, J-B Mouret, and Stéphane Doncieux. The transferability approach: Crossing the reality gap in evolutionary robotics. *Evolutionary Computation, IEEE Transactions on*, 17(1):122–145, 2013.

[16] Minh Nghia Le, Yew-Soon Ong, Yaochu Jin, and Bernhard Sendhoff. Lamarckian memetic algorithms: local optimum and connectivity structure analysis. *Memetic Computing*, 1(3):175–190, 2009.

[17] Joel Lehman and Kenneth O Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223, 2011.

[18] Hod Lipson and Jordan B Pollack. Automatic design and manufacture of robotic lifeforms. *Nature*, 406(6799):974–978, 2000.

[19] Peter Merz. Advanced fitness landscape analysis and the performance of memetic algorithms. *Evolutionary Computation*, 12(3):303–325, 2004.

[20] Pablo Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. *Caltech concurrent computation program, C3P Report*, 826, 1989.

[21] Ferrante Neri and Carlos Cotta. Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, 2:1–14, 2012.

[22] Ferrante Neri and Carlos Cotta. Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, 2:1–14, 2012.

[23] Ferrante Neri and Ernesto Mininno. Memetic compact differential evolution for cartesian robot control. *Computational Intelligence Magazine, IEEE*, 5(2):54–65, 2010.

[24] Yew Soon Ong and Andy J Keane. Meta-lamarckian learning in memetic algorithms. *Evolutionary Computation, IEEE Transactions on*, 8(2):99–110, 2004.

[25] Yew-Soon Ong, Meng-Hiot Lim, Ning Zhu, and Kok-Wai Wong. Classification of adaptive memetic algorithms: a comparative study. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 36(1):141–152, 2006.

[26] Ingo Paenke, Yaochu Jin, and Jürgen Branke. Balancing population-and individual-level adaptation in changing environments. *Adaptive Behavior*, 17(2):153–174, 2009.

[27] Jordan B Pollack and Hod Lipson. The golem project: Evolving hardware bodies and brains. In *Evolvable Hardware, 2000. Proceedings. The Second NASA/DoD Workshop on*, pages 37–42. IEEE, 2000.

[28] Matt Quinn, Lincoln Smith, Giles Mayley, and Phil Husbands. Evolving controllers for a homogeneous system of physical robots: Structured cooperation with minimal sensors. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 361(1811):2321–2343, 2003.

[29] John Rieffel, Frank Saunders, Shilpa Nadimpalli, Harvey Zhou, Soha Hassoun, Jason Rife, and Barry Trimmer. Evolving soft robotic locomotion in physx. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, pages 2499–2504. ACM, 2009.

[30] Eivind Samuelsen, Kyrre Glette, and Jim Torresen. A hox gene inspired generative approach to evolving robot morphology. In *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference*, pages 751–758. ACM, 2013.

[31] Karl Sims. Evolving 3d morphology and behavior by competition. *Artificial life*, 1(4):353–372, 1994.

[32] Karl Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22. ACM, 1994.

[33] Richard A Watson, SG Ficiei, and Jordan B Pollack. Embodied evolution: Embodying an evolutionary algorithm in a population of robots. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 1. IEEE, 1999.

[34] Darrell Whitley, V Scott Gordon, and Keith Mathias. Lamarckian evolution, the baldwin effect and function optimization. In *Parallel Problem Solving from Nature-PPSN III*, pages 5–15. Springer, 1994.

[35] Tony D. Williams. *The Penguins.* Oxford University Press, 1995.

[36] David H Wolpert and William G Macready. No free lunch theorems for optimization. *Evolutionary Computation, IEEE Transactions on*, 1(1):67–82, 1997.

[37] Juan C Zagal, Javier Ruiz-del Solar, and Paul Vallejos. Back to reality: Crossing the reality gap in evolutionary robotics. In *IAV 2004 the 5th IFAC Symposium on Intelligent Autonomous Vehicles, Lisbon, Portugal*, 2004.