Damage Recovery for Simulated Modular Robots Through Joint Evolution of Morphologies and Controllers

Submitted: 3rd September 2018; accepted: 22nd January 2019

Djouher Akrour, NourEddine Djedi

DOI: 10.14313/JAMRIS_1-2019/2

Abstract:
In order to be fully autonomous, robots have to be resilient so that they can recover from damages and operate for a long period of time with no human assistance. To be resilient, existing approaches propose to change the robots' behavior using a different control system when a hardware fault or damage occurs. These approaches are used for robots which have fixed morphologies. However, we cannot assume which morphology would be optimal for a given problem and which morphology allows resilience. In the present paper, we introduce a new approach that generates resilient artificial modular robots by evolving the robot morphology along with its controller. We used a multi-objective evolutionary algorithm to optimize two objectives at a time, which are the traveled distance of a damage-free robot and the traveled distance of the same robot with damaged parts. The result of preliminary experiments demonstrates that during evaluation, when robots are deliberately faced to motor failures, the evolution process can optimize and generate new morphologies for which the robot's behavior is less affected by damage. This makes the robot capable to recover its ability to move forward.

Keywords: *Artificial life, Controller, Evolutionary robotics, Modular robots, Resilience*

1. Introduction

Building resilient robots capable of recovering from damages is a central and challenging question [1, 2]. Making such robots would make them able to sustain their ability to pursue their missions when a hardware degradation occurs, with no human assistance. In order to have such robots, researchers propose to change robots' behavior using new control systems. Some approaches attempt to let the robot use evolutionary algorithms to learn new compensatory behavior online after detecting the damage. In this case, learning is done either on the physical robot with an embedded learning [3, 4], or using a self-modeling approach, which is based on transfers of learned behaviors between a physical simulator and the robot [2]. This approach showed its capacity to decrease the learning time. Recently, Cully et al. [1] propose to create an offline behavior repertoire. The latter is used during the mission to perform an online search of the best suitable controller for the current situation, therefore speeding-up the adaptation-to-damage process. While these approaches are used on robots with fixed morphologies, we propose to evolve the controller along with the morphology. It has been proved that co-evolving morphologies with their behavior can produce more adaptable and evolvable robots [5, 6]. In the early 90's, Sims [7] pioneered the field by co-evolving the morphology and the control of artificial creatures. Since then, many methodologies on evolution, learning and generating artificial creatures have been studied, either in simulation [8] or embedded in real world robots [9]. However, it remains a challenging task [6].

By evolving the morphology, we seek to generate robots' body plans that allow resilience without changing the controller each time a damage occurs. In other words, we want to optimize and produce robots which morphologies lower the impact of damages on their locomotion. Robots from literature, snakes [4] or multi-legged [1], have critical joints or limbs that play a central role on the motion efficiency. When failure affects one of these parts, the robot capacity to move is strongly compromised. In order to overpass this limitation, we aim at evolving robots with no such parts, and instead, focusing on robots containing joints and limbs that collaborate to compensate possible failures with the lowest possible effect on their objective.

With this aim in mind, we use a multi-objective evolutionary algorithm to evolve robots maximizing both, the traveled distance of a damage-free robot and the traveled distance of the same robot with damaged parts. Robots used in this work are modular. Their morphologies and controllers are encoded in genomes using oriented graphs. In order to evaluate the performance of our robots, we use Gazebo, a realistic multi-robot simulator often used to simulate large robots (humanoids, wheeled-robots, etc.). It is recognized that Gazebo uses simulated sensors that produce a data stream which closely matches data from real-world physical sensors. The simulator provides for the environment and the simulated objects numerous physical attributes that we can set up with realistic values.

Results of preliminary experiments described in this paper show that by intentionally damaging the robot's motors during the evaluation, the evolution process can generate robots that can adapt their motions each time a failure occurs.

The paper is structured as follows: in Section 2, we introduce the virtual modular robots and show how the morphology and the controller are co-generated. In section 3, we present how resilience can be part of the evolutionary process. Section 4 describes the experimental results. Section 5 contains a discussion. Finally, section 6 concludes with a summary of the work and some possible perspectives.

2. Virtual Robots

Robots used in the present work are modular [13]. They are composed of a set of cuboids modules that are linked by joints of equal sizes. With the aim of making simulations more realistic, interpenetration between neighboring modules is forbidden during runtime. In this work, robots morphologies and controllers are encoded in genomes using oriented graphs [7], where each node contains the phenotypic parameters of one module, such as, the size, the available sensors and a local controller that controls one of the joints connected to it. This representation allows modularity and symmetry in the generated morphologies. All details of the genetic representation are provided in [11].

Each robot has two levels of controllers: a local controller included in every single module that controls the connecting joints and a global controller able to control all robot' joints to globally modify the behavior of the robot when necessary. All controllers are a Multi-Layer Perceptrons (MLP) [14] using a Hyperbolic Tangent activation function. They contain at most 3 hidden layers which can have from 3 to 10 neurons. The input layer contains 10 neurons, receiving data from sensors and from communication. The output layer contains 3 neurons, one for communication and the two others are used to compute the torque that will be applied to the joint. The result value is then normalized to fit the range of allowed torque. Communication neurons are directly connected from the output of a module to an input of another module to leave open the emergence of communication protocol within the robot. An example of the distributed control system is illustrated in Fig. 1.

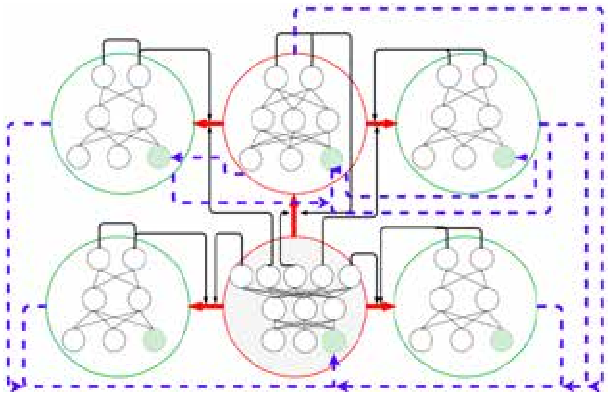


Fig. 1. The phenotype generated from the genotype shown in Fig. 2. Blue, black and red lines represent respectively the communication circuit, torque and effectors. Grey module is the root. It contains the global controller [11]

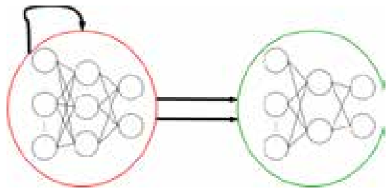


Fig. 2. A typical example of a genotype. The outer graph describes the morphology (assembly and features) while the inner graph on each node is the controller [11]

2.1. Sensors and Effectors

Sensors are used to inform robots about both the external world and the internal state. In our work, sensing capabilities are restricted to joint inner state (angle, torque), module inner state (force, orientation, linear and angular velocity) and contact sensors. Effectors apply torques onto robot's joints. The torque applied to each joint is a sinusoidal function. The first neuron of the output layer codes for the amplitude and the second for the phase.

2.2. Simulator and Realism

In order to obtain robots which behavior are as close as possible to real physical robots, simulations are using Gazebo with interpenetration disabled. Gazebo is a popular framework to simulate robots. It allows the production of a data stream which closely matches data from real-world physical sensors. In addition, it provides numerous physical attributes for the environment and the simulated objects. They are set up as follow:

- *Module dimensions*: they are ranging from 5 cm to 10 cm.
- *Module mass*: the mass is calculated from the volume of the module and its density. Modules have the density of water.
- *Module moment of inertia*: it determines the difficulty of making an object rotate. It is the most influential parameter on the behavior realism.
- *Friction*: when applied on module surface, opposing frictional force can avoid robot sliding, and when applied in joints it will avoid the unrealistic vibration moves. We set joint and module surface friction respectively to 0,2 and 0,5.
- *Joint damping:* depending on joint velocity, damping allows the energy dissipation. This can avoid bouncing movements. It is set to 0,02.
- *Joint torque*: it is the required force that can make a joint rotate and raise all the modules attached to him without breaking up the joint. However we set a torque limitation to 1,75 Nm.
- *Joint velocity*: the maximum velocity allowed to a joint is 5 rad/s.

3. Resilience Approach

After technical failures, resilient robots must keep moving forward. Changing the direction of motion after damage may produce non-desirable results and lead to a task failure, such as in robot surveillance, where robots have to maintain specific trajectories. Therefore, in our opinion, efficacy and velocity of the locomotion are of lower importance than keeping a given direction.

Robots can endure several kinds of damages such as motor failures, missing or broken limbs (legs), etc. In order to reduce the complexity of the recovery process, we focus in this study on motor failures, i.e. when a joint is disabled. We also limit during evaluation to one joint failure.

3.1. Objectives

The objective is to produce either robots which morphologies allow for resilience or robots capable of finding new locomotion strategies to recover from damage. In other words, we aim at evolving robots whose locomotion efficiency is not determined by specific joints and limbs. So the failure of any motor that makes limbs uncontrolled and inactive will not substantially affect the locomotion efficiency.

3.2. Performance Function

Robots are evaluated in the environment with and without faulty joints. Each evaluation takes 10 s of a simulation time. The initial point is taken after the stabilization of the robot (after 2 s). So, the overall evaluation time of a robot depends on the number of its joints. To evolve robots on which faults will not affect the locomotion, especially the direction, robots are evaluated on two objectives: (1) maximize the traveled distance with no failure in any direction (Eq. 1) (2) maximize the traveled distance with a faulty joint in the direction given by objective (1). The second objective can be rewritten as the minimization of the distance between the point reached by the damaged robot and some target point set in the direction given by objective (1). Among all motor failures evaluations, we choose to take the highest remaining distance as presented in Eq. 2. We note that the goal of the robot is to maintain the same direction of locomotion, not to reach the goal. So the remaining distance is a measure that let us know robots that sustain the direction of locomotion and keep moving forward.

$$dis = \sqrt{\left(f_x - i_x\right)^2 + \left(f_y - i_y\right)^2} \tag{1}$$

$$rem = \max_{\substack{j:\,indi\ of\ the \\ faulty\ joint}} \left\{ \sqrt{\left(o_x - f_x^j\right)^2 + \left(o_y - f_y^j\right)^2} \right\} \tag{2}$$

where *i* and *f* are respectively the initial and the final positions of the robot during evaluation, *o* is the target that must be reached by the robot with a faulty joint. It is 1 m far from the starting point (Eq. 3). In that way we ensure that the locomotion of the robot is forward even if there are failures.

$$\left\{ \begin{array}{l} o_x = \mathrm{i}_x + 1/\mathrm{dis}\left(f_x - i_x\right) \\ o_y = \mathrm{i}_y + 1/\mathrm{dis}\left(f_y - i_y\right) \end{array} \right\} \tag{3}$$

3.3. Joint Evolution

In order to test and evaluate the efficiency of our system, we conducted a set of experiments. Each experiment starts with a random generation of 80 robots. Experiments ran for 400 generations. Crossover and mutation rate were respectively 35% and 75%. The robots' behavior we are aiming at requires to optimize more than one objective. Having several objectives to satisfy in a single run needs the use of Multi-objective Optimization Evolutionary Algorithm (MOEA). Instead of searching one optimal solution, we produce a set of multiple trade-off solutions from which one solution can be selected from the decision maker. In this context, we use the Non-dominated Sorting Genetic Algorithm (NSGA-II) [15], which has been successfully applied to solve multi-objective problems.

We optimize then the traveled distance of a damage-free robot and the traveled distance of the same robot with damaged parts. However, we notice that it is difficult to find robots that go forward after damage and travel also a long distance. In order to overcome this problem, we introduce a penalty that makes a pressure to favor the appropriate robots' morphologies and controllers to emerge. In other words, we reduce the resilience of robots that have a fitness less than 20 cm (traveled distance when no motor failure occurs), we multiply the value of resilience (the remaining distance) by 1.5.

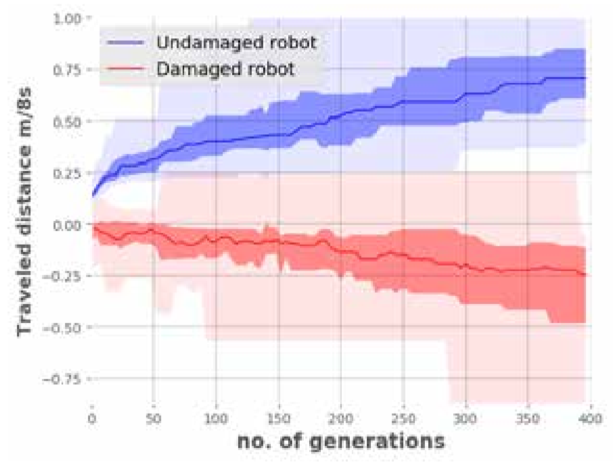
In addition, when we introduce damage during the evaluation, we inactivate all the joints of the robot successively and assess each motor failure separately. This is necessary because if we select only some of them, the process of evolution will favor robots whose joints selected for damage are not important. For instance joints such us neck and fingers. And this is due to the fact that the sole purpose is to satisfy the objective function even if it does not meet the expectations and the final goal of engineers. Therefore, if a motor that controls a joint of an important role fails during mission, the robot remains in place.

4. Experimental Results

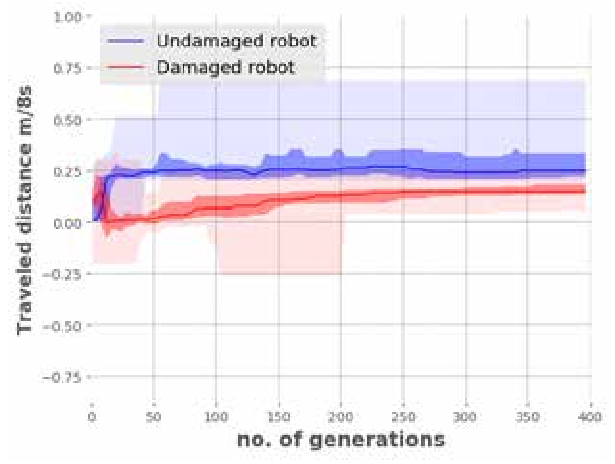
Graphs in Fig. 3 illustrate the convergence curve of the evolutionary process. Plots show the median, the interquartile range (dark ruban), the min and max fitness (light ruban) of 25 independent evolutionary runs.

Graph Fig. 3a represent the best solutions according to the first objective (robots that can travel a long distance when there are no failures), while Fig. 3b represent the best solutions according to the second objective (robots that can travel a long distance toward target when there are failures). Graphs of the figure demonstrate the distance traveled by robots with and without failure at each generation.

Fig. 3a shows that, on average, robots can travel for up to 70 cm, but in counterpart, when a damage occurs, they either remain in place or keep moving in another direction (negative distance) for about 25 cm. Concerning the solutions taken from the other extremity of Pareto Front (Fig. 3b), intact robots travel in average 25 cm and when damage occurs, robots travel in average 15 cm. However, the displacement is toward target like shown in Fig. 3b.



(a)



(b)

Fig. 3. Solutions taken from extremities of Pareto front; best resilient robots (b) and the less ones (a) that can travel long distances without damage. Blue curve represent the traveled distance without faulty joints. Red curve represent the traveled distance with a faulty joint

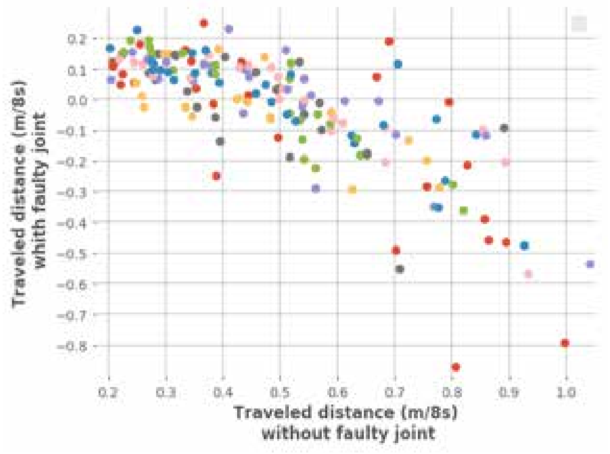


Fig. 4. Individuals from Pareto Front of last generation of the 25 evolutionary runs

Fig. 4 illustrates the last generation Pareto fronts of the 25 evolutionary runs. Robots that are in the left side are the more resilient ones. When a failure occurs, they always maintain the same direction of locomotion, whereas others do not. We can notice that robots that travel from 20 cm to 40 cm, when damaged, have the capacity to maintain the same direction of locomotion and almost can achieve half the distance in the same period of time.

5. Discussion

We can see that robots that travel slowly can be easily resilient, while those that go fast traveling more distance are not. These robots have difficulties to maintain the direction of displacement when damage occurs. The generated morphologies in this case are interesting and show good strategies of locomotion. Nevertheless, there is always one or two joints that play the crucial role for doing effective moves. Therefore, the robot can either remain in place or completely change the direction of locomotion.

Concerning robots that go slow and travel less distance, they are able to keep the same direction of displacement after damage. We have noticed that the failure of any motor does not affect drastically the manner of locomotion. The performance, in the worst case, drops to 40%. The control system succeeded at controlling all the joints of a robot in order to cooperate when there is a faulty joint. This is because none of the joints plays a crucial role for doing the effective moves. However, in return robots travel slowly.

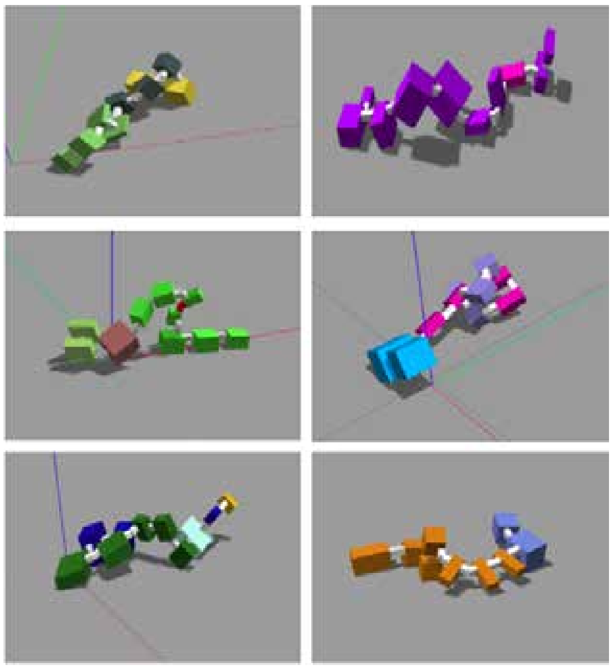


Fig. 5. Examples of modular robots that are resilient

Obtained morphologies of these resilient robots (Fig. 5) look in general the same in each run. We can notice that they are snake-like but with different joint rotation axis. Having this kind of joints allows robots to develop crawling behavior. Each time one joint is inactivated, other joints can compensate and continue the displacement. Resulting robots are capable of displacement on the same line and direction with and without damage, even though their performance (velocity) can be slightly reduced. However, if these robots had more evaluation time, they may reach the same positions as when there was no damage.

6. Conclusion

This paper seeks to investigate whether resilience can be emerged by evolving the robot's morphology or not. We deliberately caused damage to robot motors during evaluation time. We have noticed that the evolution can generate morphologies that cannot be considerably affected by the damage. Our approach was able to generate interesting robots locomotion as well as an ability to recover from damage and keep moving forward. The same controller was then able to control the intact robot and also the robot with damage. Since robots can be affected by several kinds of damage, we will on a future work induce more extreme damages such as missing or broken limbs.

AUTHORS

Djouher Akrour* – Department of Computer Science, Biskra University, 07000, Algeria,
e-mail: akrour_djouher@yahoo.fr.
NourEddine Djedi – Department of Computer Science, Biskra University, 07000, Algeria,
e-mail: noureddine.djedi@gmail.com.

*Corresponding author

REFERENCES

[1] A. Cully, J. Clune, D. Tarapore and J.-B. Mouret, "Robots that can adapt like animals", *Nature*, vol. 521, no. 7553, 2015, 503–507
DOI: 10.1038/nature14422.

[2] J. Bongard, V. Zykov and H. Lipson, "Resilient Machines Through Continuous Self-Modeling", *Science*, vol. 314, no. 5802, 2006, 1118–1121
DOI: 10.1126/science.1133687.

[3] D. Berenson, N. Estevez and H. Lipson, "Hardware evolution of analog circuits for in-situ robotic fault-recovery". In: *2005 NASA/DoD Conference on Evolvable Hardware (EH'05)*, 2005, 12–19
DOI: 10.1109/EH.2005.30.

[4] S. H. Mahdavi and P. J. Bentley, "Innately adaptive robotics through embodied evolution", *Autonomous Robots*, vol. 20, no. 2, 2006, 149–163
DOI: 10.1007/ s10514-006-5941-6.

[5] J. C. Bongard, A. Bernatskiy, K. Livingston, N. Livingston, J. Long and M. Smith, "Evolving Robot Morphology Facilitates the Evolution of Neural Modularity and Evolvability". In: *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, New York, NY, USA, 2015, 129–136
DOI: 10.1145/2739480.2754750.

[6] N. Cheney, J. Bongard, V. SunSpiral and H. Lipson, "Scalable co-optimization of morphology and control in embodied machines", *Journal of The Royal Society Interface*, vol. 15, no. 143, 2018
DOI: 10.1098 /rsif.2017.0937.

[7] K. Sims, "Evolving Virtual Creatures". In: *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, New York, NY, USA, 1994, 15–22
DOI: 10.1145 /192161.192167.

[8] N. Lassabe, H. Luga and Y. Duthen, "A New Step for Artificial Creatures". In: *2007 IEEE Symposium on Artificial Life*, 2007, 243–250
DOI: 10.1109/ ALIFE.2007.367803.

[9] A. E. Eiben and J. Smith, "From evolutionary computation to the evolution of things", *Nature*, vol. 521, no. 7553, 2015, 476–482
DOI: 10.1038/ nature14544.

[10] C. C. Coello, G. B. Lamont and D. A. van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Genetic and Evolutionary Computation Series, Springer US, 2007
DOI: 10.1007/ 978-0-387-36797-2

[11] D. Akrour, S. Cussat-Blanc, S. Sanchez, N. Djedi and H. Luga, "Joint evolution of morphologies and controllers for realistic modular robots". In: *22nd Symposium on Artificial Life and Robotics (AROB 2017)*, Beppu, Japan, 2017, 57–62.

[12] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator". In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004, 2149– 2154
DOI: 10.1109/IROS.2004.1389727.

[13] M. Yim, W. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins and G. S. Chirikjian, "Modular Self-Reconfigurable Robot Systems [Grand Challenges of Robotics]", *IEEE Robotics Automation Magazine*, vol. 14, no. 1, 2007, 43–52
DOI: 10.1109/MRA. 2007.339623.

[14] M. T. Hagan, H. B. Demuth and M. H. Beale, *Neural network design*, Boston: PWS Pub, 1996.

[15] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II", *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, 2002, 182–197
DOI: 10.1109/4235.996017.