Optimal design of modular soft body crawling robot structure based on genetic algorithm

Changxin Guo

Columbia University NewYork, United States

cg3282@columbia.edu

Abstract. This paper explores how a soft robot can be designed and built with a limited material framework to optimize evolutionary iterations to achieve the fastest crawling configuration. The limited framework of the soft robot is based on a modular design, which divides the soft robot into several cubes composed of masses and springs. The morphological self-optimization of the modular soft robot is controlled by set variable parameters combined with an incentive mechanism and an evolutionary algorithm.

Keywords: Evolutionary Algorithms; Modularity; Soft robot; Crawling Robot.

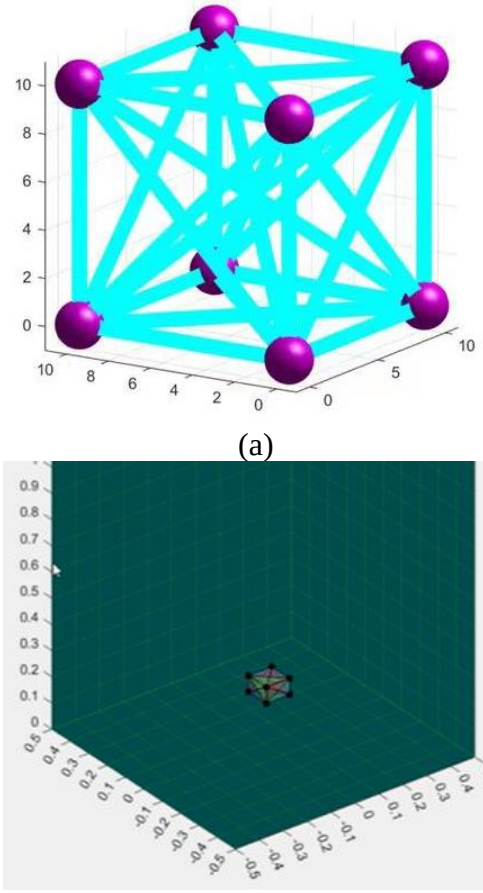
1. Introduction

According to Karl Sims, computational evolution can produce interesting forms that resemble natural creatures, and an evolutionary algorithm was used to evolve a virtual creature to demonstrate that the algorithm can evolve a "creature" into such a complex form that it resembles nature [2]. In Hod Lipson's research, it is argued that the above mentioned research area seems to have reached the upper limit of complexity, and most of the works are composed of rigid elements that limit the design space [1]. Therefore, the current breakthrough in this field requires breaking the rigid design by using soft material robots for simulation. By eliminating the traditional manufacturing constraints, new design spaces can be created to better fit natural systems and achieve better genetic algorithm results [3]. Also we learned that algorithms such as the MOLE algorithm of Mouret and Clune [5, 9] or the algorithm of Lehman and Stanley [6], also have the potential to create an extremely complex and diverse set of morphologies and behaviors. Moreover, the approach we are currently seeking is a solution to achieve optimal speed within a limited material and a limited design space, thus reinforcing the need for a design idea that combines a complete evolutionary algorithm with a modular design. Soft robots have shown potential in several areas of robotics, and soft robots have excellent grasping performance [4] and human-robot interaction [10]. There are many papers on rigid body robots [7] and few attempts to evolve by soft materials [8], so exploring the most efficient structural design for soft robots to achieve some purpose with limited space and limited materials is very promising and promising direction.

2. Simulate individual cube modules in realistic physical conditions

The first stage is to define the composition of a single cube module —— each cell consists of eight vertex masses and two connected springs. For each mass object the following properties are required in order to meet realistic physical conditions: (i) mass: m [kg]; (ii) position length vector: p [m]; (iii) velocity vector: v [m/s]; (iii) acceleration vector a [m/s^2].And a realistic physical spring must contain the following properties: (i) spring constraint: k [N/m]; (ii) original rest length of the spring itself: L0 [m]; (iii) masses of the two objects connected to the spring: m1, m2 [kg].After defining the parameters required to build a single cell module, the next step is to create an initial cube for testing, as in Figure 1(a).Next, it is placed into a simulated realistic physical environment for free fall to test whether its physical state meets expectations, so we need to add additional parameters that meet the physical conditions: (i) the gravity constant vector: g=(0,0,-9.81) [m/s^2]; (ii) the height of fall;(iii) Kinetic coefficient of friction; (iiii) Damping Constant; (iiiiii) Ground restoration constant [N/m].And some necessary parameters: (i) Time step of simulation calculation: dt [s]; (ii) Global start time: T=0

[s]. In this way, the effect of a "breathing" cube free fall can be simulated by Matlab, as shown in Figure 1(b).



(a)

(b)

Figure 1. Simulation of individual modules: (a) Create an initial cube; (b) Simulation of a "breathing" cube free fall.

3. Simulate evolvable individual cube modules under realistic physical conditions

3.1. Description of the parameters for actuation pattern

In the previous stage, a "breathing" cube was simulated in Matlab to test the effect of free fall. The parameters we used for the physics simulator are: Timesteps dT = 0.0005s; Kinetic coefficient of friction = 0.6; Damping Constant = 0.99.

Due to the arithmetic limitations of graphical calculations, a complex model cannot be used for simulation, so using a single cube as the basic shape of the robot. Here the movement of the robot is achieved by changing parameters such as the initial length of each spring of the cube species and the spring elasticity coefficient. Each cube robot consists of a combination of 8 masses and 28 springs. Therefore, each spring of each cube will have 28 different sinusoidal functions of the form. L0=a+b*sin(wt+c). In addition, it was previously mentioned that the spring's elasticity coefficient k is also a variable.

Also due to the limitations of the computing power of graphical computing, it is not possible to simulate the coefficients in the function of the spring randomly selected, so when performing the genetic algorithm simulation, the problem of genetically encoding the parameters of the function of the spring can use four specific choices of parameters, each spring is determined by the following coefficients [k,b,w,c], then set four options to choose from:[1000,0,1,0] [2000,0,1,0] [5000,0.05,1,0] [5000,0.05,1,0].

3.2. Description of the parameters for evolutionary process

During the evolution process, there will be 30 variable settings for each evolution, and these 30 variable settings will be imported into 30 different robots. Each robot will run for a calculated 1000 evolutionary cycles. The cube velocity at different variables is found by calculating the net displacement of the center of mass divided by the total run time. There are generally two traditional selection methods to quickly decide whether a variable is selected or retained: namely, the tournament ranking method and the roulette wheel ranking method.

For the selection of offspring, dividing the 30 results into three groups, and each group will select the parameters of the cubes with the top 20% speed to be inherited in the next generation. By doing this way, it is possible to avoid being trapped in a locally optimal solution. Next crossover is performed and the probability of crossover is set to 24/30. for mutation, the mutation rate will be set to 0.2.

3.3. Description of the crossover and mutation process for evolutionary process

Taking out a list of parameters of the same kind for each spring variation, for example, about k, as k1, k2, . . k28, each k is randomly assigned to a value as the father, and after randomly assigning again this list as the mother, two index positions are randomly generated in the set of variables. The set of variables from 0 to the first index position will be from the parent set, from the first index to the second index will be from the parent set, and from the second index to the last index will be from the parent set again. By this multi-point crossover, the diversity of results can be increased.

Regarding mutation, it is a random selection of 2 parameters from this list of 28 parameters to be changed when mutation occurs to regenerate the new values. This avoids the homogeneity of the results and thus increases the diversity.

Figure 2 illustrates the learning curve.

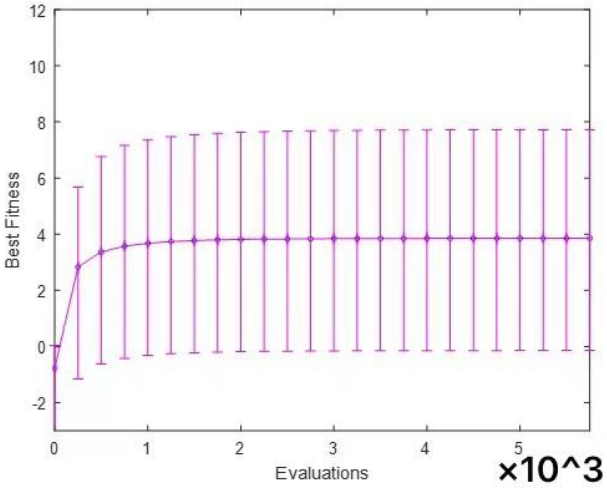


Figure 2. Learning curve of crawling speed.

4. Simulating evolvable soft-bodied crawling robots in realistic physical conditions

At the third phase, it is time to evolve a controller for a robot with variable morphology. First, we follow the program operations of the cube based robot in last phase, and on this basis we make the robot have variable morphology during the evolution by changing the morphology of the cube. The robot is shown in the figure 3 below. In this part, the main goal is to encode classes for building the robot's deformable and implementing evolutionary algorithms.

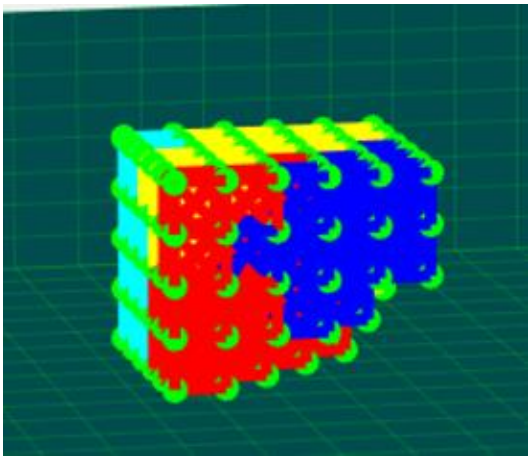


Figure 3. Soft crawling robot with variable morphology.

4.1. Description of the parameters for simulation

In this phase, we use the physical simulator created in last phase. The parameters we used for the physical simulator are: Timesteps dT = 0.0005s; Kinetic coefficient of friction = 0.6 ; Damping Constant = 0.99; Ground restoration constant=10000N/m; Gravity= 9.81N/kg.

4.2. Description of the parameters for variable morphology robot

Due to the limitations of the computing power of graphics computing, it is not possible to use well-rendered surfaces and ground shadows for simulating soft robots, so using a cube with only masses and springs rendered as the basic shape of the robot, as show in Figure3. Here, the motion of the robot is achieved by varying parameters such as the initial length of each spring of the cube species and the spring elasticity coefficient. Each cube robot consists of a combination of 8 masses and 28 springs. Therefore, in order to meet the requirements of a deformable robot, we set that the springs in each cube will have the following parameters taken as: hard spring, soft spring, sine function, neg sine function, and empty. example: where the soft and hard springs change the magnitude of the spring coefficient k, the sine function takes the form L0=a+b*sin(wt+c). In addition, empty is where the mass of some cube and the spring disappears, thus changing the shape of the robot.

And again, due to the arithmetic limitations of the graphical calculations, it is not possible to simulate the random selection of coefficients in the spring function, so when performing the genetic algorithm simulation, the problem of genetic coding of the parameters of the spring function giving four specific parameter choices。Set lables as 'Neg Sine','Sine','Soft','Hard'.

Summary Robot Parameters:Mass=0.1kg; Length of tetrahedron=1m; Soft spring constand=500N/m; Hard spring constand=2000N/m; Gene a range=-1--1; Gene b range= -π/ 2--π/ 2; Gene c range= -2*π -- 2*π.

4.3. Description of the parameters for evolutionary process

In the evolutionary process, the several modular cubes in the soft robot were previously divided equally into five forms of variables, lables are:hard spring, soft spring, sine spring,neg sine spring, where empty is the elimination of this cube, and then further evolve the parameters of the sine neg sine spring, each evolution will have 25 variable settings, and these 25 variable settings will be imported into 25 different robot populations. Each robot will run the computation for 100 evolutionary cycles. We calculate the cube speed for the different variables by dividing the net displacement of the center of mass by the total run time.

The parameters that will allow the robot to crawl faster are next selected for retention, and there are generally two traditional methods of selection: namely, the tournament ranking method and the roulette wheel ranking method. For the progeny selection, dividing the 25 results into some groups and each group will select the parameters of the top 50% of the speed cubes to be inherited in the next

generation. In this way, it is useful to avoid being trapped in a local optimal solution. Next crossover and mutation are performed. Where the mutation rate will be set to 0.02. In addition, the proportion of random individuals added to the population every gen set as 0.12.

The evolutionary parameters are as follows: Population size=25; Generations number=100; Selection pressure=0.5; Proportion of children that get mutated=0.02; Proportion of random individuals added to the population every gen= 0.12

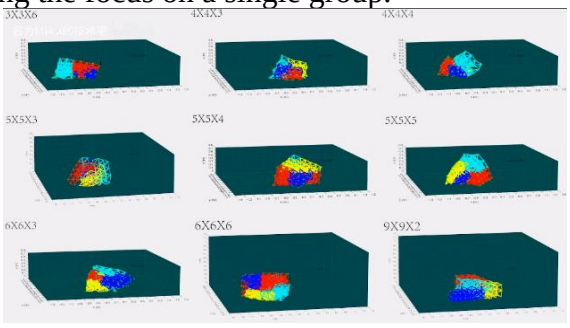
4.4. Description of the crossover and mutation process for evolutionary process

If the label is not empty, then we come up with a list of four of the same kind of parameters for the springs in each cube: 'Neg Sine', 'Sine', 'Soft', 'Hard' respectively, and the springs in each cube are randomly assigned a value as the parent set, and after randomly assigning this list again as the parent set, two index positions are randomly generated in the set of variables. The set of variables from 0 to the first index position will come from a parent set, from the first index to the second index will come from another parent set, and from the second index to the last index will again come from the previous parent set. By this multi-point crossover, the diversity of results can be increased.

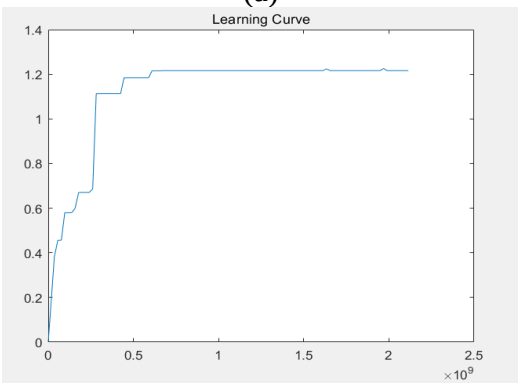
Regarding mutation, it is a random change of the label of some cubes when mutation occurs, so that all the springs in this cube are reselected among the five labels 'Neg Sine', 'Sine', 'Soft' and 'Empty'. This avoids homogeneity of the results and thus increases the diversity.

5. Result

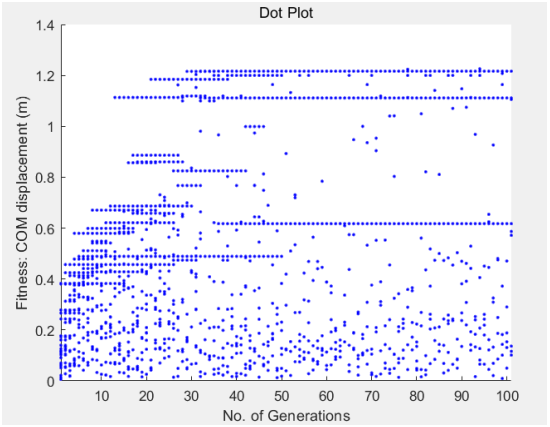
Using the mentioned method for simulation testing, firstly, the upper limit of the module containing the length, width and height of the soft robot was determined first, and on this basis, the simulation was conducted separately. As shown in Fig. 4(a), there are nine groups tested, and the maximum values of the length, width, and height of each soft robot are divided into 3X3X6; 4X4X3; 4X4X4; 5X5X3; 5X5X4; 5X5X5; 6X6X3; 6X6X6; 9X9X2. By calculating the crawling displacement of the center coordinates of the nine groups of soft robots in the same time, we can find out for each group the evolved new configuration's The fastest crawling speed of each new evolved configuration was found. The learning curve in Fig. 4(b), the Populating dot plot in Fig. 4(c) and the Populating Diversity chart in Fig. 4(d) are used to observe the evolution of crawling speed as the number of iterations increases by placing the focus on a single group.



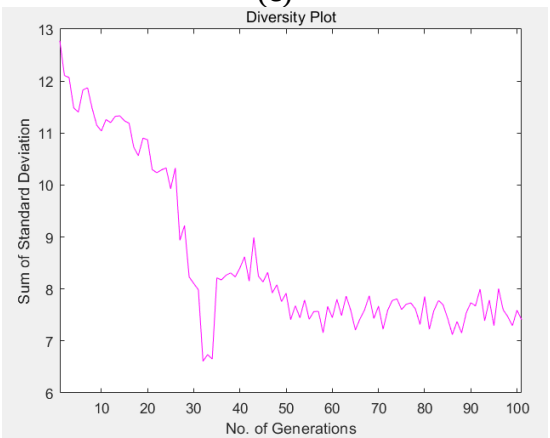
(a)



(b)



(c)



(d)

Figure 4. The images shown are: (a) The evolution of nine groups of soft robots; (b)Learning curve of evolutionary effect with increasing number of iterations; (c)Populating dot plot of evolutionary effect with increasing number of iterations; (d)Populating Diversity chart of evolutionary effect with increasing number of iterations.

However, one of the drawbacks of the evolutionary approach mentioned in the paper is that the modularity is not refined to a single mass and a single spring, so as to increase the complexity and diversity of the soft robot form that can evolve and iterate, instead of being limited to a cubic block for each module to be combined and changed. More detailed research can be conducted in this direction in the future.

References

[1] Cheney, N., MacCurdy, R., Clune, J. and Lipson, H., 2014. Unshackling evolution: evolving soft robots with multiple materials and a powerful generative encoding. ACM SIGEVOlution, 7(1), pp.11-23.

[2] K. Sims. Evolving virtual creatures. In Proc. of the 21st Annual Conf. on Computer Graphics and Interactive Techniques, pages 15–22. ACM, 1994.

[3] Lipson, H., 2014. Challenges and opportunities for design, simulation, and fabrication of soft robots. Soft Robotics, 1(1), pp.21-27.

[4] S. Hirose and Y. Umetani. The development of soft gripper for the versatile robot hand. Mechanism and MachineTheory, 13(3):351–359, 1978.

[5] J.-B. Mouret and J. Clune. An algorithm to create phenotype-fitness maps. Proc. of the Artificial Life Conf., pages 593–594, 2012.

[6] J. Lehman and K. O. Stanley. Evolving a diversity of virtual creatures through novelty search and local competition. In Proc. of the Genetic & Evolutionary Computation Conf., pages 211–218. ACM, 2011.

[7] S. Nolfi, D. Floreano, O. Miglino, and F. Mondada. How to evolve autonomous robots: Di↵erent approaches in evolutionary robotics. In Artificial Life IV, pages 190–197. Cambridge, MA: MIT Press, 1994.

[8] J. Rieffel, F. Saunders, S. Nadimpalli, H. Zhou, S. Hassoun, J. Rife, and B. Trimmer. Evolving soft robotic locomotion in PhysX. In Proc. of the Genetic & Evolutionary Computation Conf., pages 2499–2504. ACM, 2009.

[9] J. Clune, J.-B. Mouret, and H. Lipson. The evolutionary origins of modularity. Proc. of the Royal Society B, 280(20122863), 2013.

[10] Article in a conference proceedings:

[11] S. Sanan, J. Moidel, and C. G. Atkeson. A continuum approach to safe robots for physical human interaction. In Int'l Symposium on Quality of Life Technology, 2011.