Design of a bio-inspired controller to operate a modular robot autonomously

Henry Hernández[1] (✉), Rodrigo Moreno[2], Andres Faina[3], and Jonatan Gomez[4]

[1,2,4]Faculty of engineering, Department of computer and industrial engineering, National university of Colombia, 110811 Bogotá D.C., Colombia,
{heahernandezma, rmorenoga and jgomezpe} @unal.edu.co
[3]Department of Computer Science, IT University of Copenhagen, 2300 Copenhagen, Denmark,
anfv@itu.dk

Abstract. A modular robot can be reconfigured and reorganized to perform different tasks. Due to the large number of configurations that this type of robot can have, several types of techniques have been developed to generate locomotion tasks in an adaptive manner. One of these techniques transfers sets of parameters to the robot controller from a simulation. However, in most cases the simulated approach is not appropriate, since it does not take into account all physical interactions between the robot and the environment. This paper shows the design of a flexible controller that adapts to the different configurations of a modular chain-type robot, which coordinates the movements of the robot using a Central Pattern Generator (CPG). The CPG is integrated with an optimization algorithm to estimate sets of movements, which allow the robot to navigate in its environment autonomously from the information of sensors and in real time.

Keywords: Genetic algorithm · Autonomous operation · Modular robot.

1 Introduction

The environmental or terrain conditions limit the access that people have to certain areas, since they can convert the activity to be developed into a high-risk one. Consequently, various robots have been proposed to reduce the accident rate, because it is possible that they adapt to unknown environments and communicate with the operator.

Some proposed robotic prototypes have been adjusted according to the terrain variability [15]. This variability of the terrain has allowed the authors to fabricate mechanisms that allow the robot to have stability in diverse environments. Among the mechanisms manufactured are the; legs, tracks or wheels. However, they still have limitations. For example, robots with caterpillars or smooth wheels cannot recover their orientation in case of capsizing [2, 16].

A partial solution to this limitation has been the development of modular robots, which have been used to reproduce patterns of animal locomotion from

body movements. A modular robot is a set of two or more coupled structures called modules. The modules can be grouped in different configurations and generate movement patterns such as: rolling, walking or crawling [13, 14, 23].

The movements generated by a modular robot have been estimated using several techniques, among which the Central Pattern Generators (CPG) stand out. An advantage of the CPG is that it allows generating movement sets in modular robots with arbitrary structures easily, since they can be represented by simple mathematical expressions [8, 17, 24].

These approaches have allowed us to estimate the movements of a modular robot using simulators, which emulate certain features of the terrain or the robot [7]. In addition, in some cases interfaces are designed that establish a link with the real robot, to transfer to the robot sets of parameters that allow it to coordinate its modules and thus generate different sets of movements [9, 11].

These sets of movements depend on the amount of degrees of freedom that the robot can have. Consequently, the dimensions of the search space do not have a certain size, increasing the difficulty in designing control mechanisms [4, 10, 19–21]. However, different control techniques have been proposed that allow this type of robot to perform tasks in unimpeded environments [1, 5].

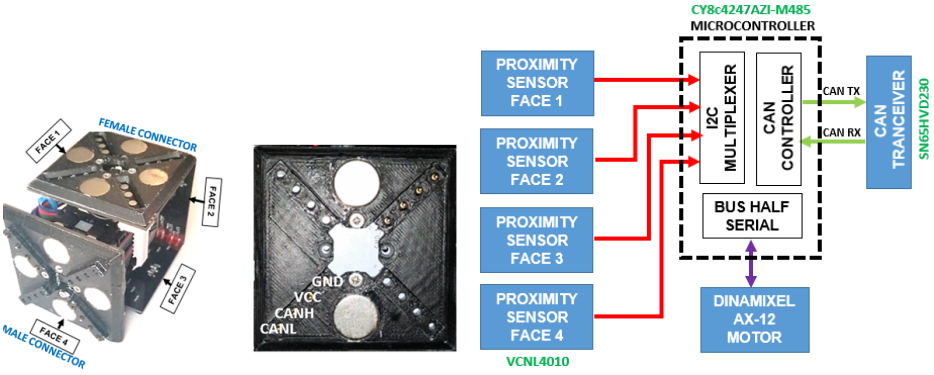
The control techniques have certain limitations, one of them is that the robot cannot adapt to irregular terrains or obstacles, from the information of sensory perception reducing its autonomy. This article proposes a partial solution to this limitation, through the development of a centralized controller that allows a modular robot to generate coordinated movements in an autonomous and adaptive way. These movements are generated by modulating the parameters of a CPG with a Genetic Algorithm (GA), which is updated from the information of the robot's sensors.

The controller was implemented in the EMeRGE (Easy Modular Embodied Robot Generator) modular robot, which is described in section 2. The adapting strategy and the control system are shown in section 3. The experimental configuration and the results are presented in section 4 and, finally, the discussion in section 5.

2 The EMeRGE modular robot

The EMeRGE modular robot was used to perform experimental tests [12]. The structures are assembled with homogeneous modules (Fig. 1a), which are connected using magnets in mating connectors. The connectors are on the four (4) sides of the module, of which three (3) of them have a female connector and the remaining has a male connector.

In addition, each module has a local driver that allows it; communicate with other modules or devices using the CAN (Controller Area Network) protocol, control the angular position of the motor and detect obstacles with four (4) proximity sensors located on each side of the module. These actions are performed by different electronic elements, which are connected by a printed circuit to a micro-controller (Fig. 1b).



(a) The EMeRGE module

(b) Diagram of the electronic circuit of each module.

Fig. 1: Structure of an EMeRGE modular robot module

The printed circuit is divided into four (4) parts that are connected to each other (Fig. 1a), which are under each face of the module to allow its connection with other modules through spring pins and pads. When connecting the modules, a four (4) wire bus is established, of which two (2) are used to transmit information using the CAN protocol and the remaining two (2) are used to energize the local controller with an external source. 12 V.

Each pin of the four (4) wire bus is flexible and allows connecting other devices to the robot that interact with it. This feature allowed coupling two accessories to the robot (Fig. 2); the first is an XBEE communication module that functions as a CAN sniffer and sends all the data shared by the modules to a computer. The second is an ultrasound sensor that allows you to measure the distance between the robot and an obstacle. In addition, this accessory works as a centralized controller that modulates the movement parameters generated by the local controllers.

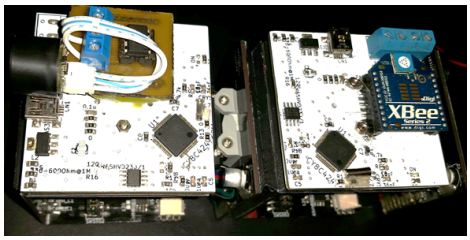


Fig. 2: Modular robot accessories EMeRGE

The link between the local controllers of the robot and the centralized controller is stable once each module has executed an initialization routine. This routine has the following functions: assign a different address to each module, detect the status of the electronic components and assign initial conditions to the movement parameters. In case the link is not established, the centralized controller will not initiate the optimization routine, which will not allow the movement parameters of the robot to be modulated.

3 CPG model used

A CPG is a model that resembles the behavior of a set of specialized neurons and one of its functions is to imitate rhythmic movements [3]. In this project, the CPG model based on coupled oscillators was implemented, which establishes a way to couple the independent outputs of the equations 1, 2 and 3 [4, 6, 9].

$$\ddot{r_i} = a_r(\frac{a_r}{4}(R_i - r_i) - \dot{r_i}) \tag{1}$$

$$\ddot{x_i} = a_x(\frac{a_x}{4}(X_i - x_i) - \dot{x_i}) \tag{2}$$

$$\theta_i = x_i + r_i cos(\phi_i) \tag{3}$$

These equations are used to estimate an approximate value of the angular position of each module (θ_i), which depends on the parameters; amplitude (r_i), phase (x_i) and offset (ϕ_i). Each independent output (θ_i) is shared with the neighboring modules and linked to its output value by means of the coupling equations 4 and 5.

The equation 4 ensures that the movement of the modules converges to a phase difference (φ_{ij}), where ϕ_i is the independent output of the module whose intrinsic oscillation depends on the value w_{ij} and ϕ_j represents the output of the neighbor module. This equation expands from the current module i to the number of neighbors j.

$$\dot{\phi_i} = w_i + \sum_{i}^{j}(w_{ij} Sin(\phi_j - \phi_i + \varphi_{ij}) \tag{4}$$

$$\theta_i^{\inf} = X_i + R_i * cos(w_i t + i\varphi_{ij+}\phi_0) \tag{5}$$

The equation 5 is a representation of the output of all the modules of the robot when they converge to an oscillating and stable state, whose amplitude R_i, phase ϕ_0 and offset X_i parameters are given by the centralized controller. Finally, the equations 1, 2, 3 and 4 are solved in each local controller using the Euler method with a step time of 300mS once the parameters of the equation 5 have been established.

3.1 CPG Optimization

An optimization technique is a method that is responsible for finding the best value in a set of solutions. Some of these techniques are based on iterative methods that evaluate the fitness value of different individuals and thus select the best. Taking into account that the fitness value is a measure, which indicates the performance of an individual when trying to solve a problem.

In this work a comparison of three optimization techniques is made, which were implemented in a centralized controller and generate sets of movements that allow a modular robot to move in its environment in an adaptive way. The characteristics of the fitness function, the individuals and the techniques implemented are described below.

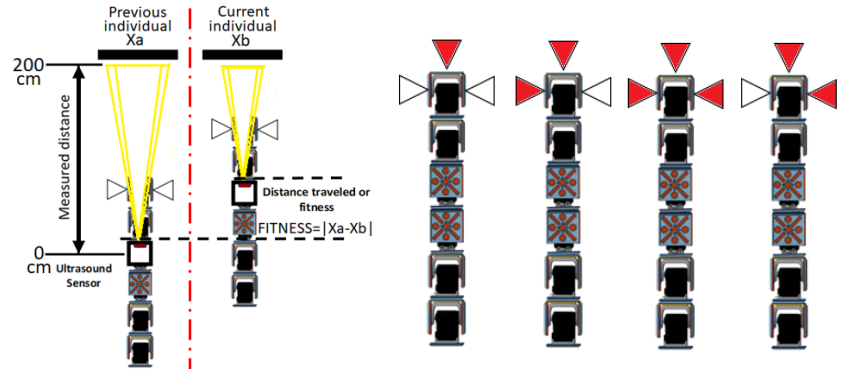
Characteristics of fitness function: The function to be optimized in this case is the distance traveled (F) by the robot (Fig. 3a), which is shown in the equation 6. From this equation the values X_a and X_b are the measurements made by the ultrasound sensor before and after modifying the parameters of the CPG. Each time the centralized controller will perform a measurement, the robot remains motionless for five (5) seconds for the sensor to stabilize.

$$F = |X_a - X_b| \tag{6}$$

The execution of the parameters of the CPG last 30 seconds, that is to say, each individual is executed during 40 seconds. In this case, the parameters of the CPG are the individuals to be evaluated and are composed of 5 different parameters, which are: Amplitude and offset of the modules according to their orientation and phase. The modules have two possible orientations, which are determined by the proximity sensor of the third face. When the sensor is active during robot initialization this module will have horizontal orientation, otherwise it will be vertical orientation.

When generating an individual, its components are limited according to their orientation and depend on a randomly generated number. If The generated number is greater than 0.5, the ranges of the components are; $0 < r_v < 1.0$, $0 < r_h < 0.2$, $0 < \phi < 2\pi$, $-0.2 < x_v < 0.2$ and $x_h = 0.0$. In another case they will be; $0 < r_v < 0.2$, $0 < r_h < 1.0$, $0 < \phi < 2\pi$, $x_v = 0.0$ and $-0.2 < x_h < 0.2$ (the subscript indicates orientation). Finally, each component is generated randomly within the aforementioned ranges and when an individual is sent to the robot each module has a filter, to classify the information and thus determine which components of the individual correspond to their CPG parameters.

Mutation of an individual: A mutation is a change that occurs in an individual, to modify their fitness value. In this work, the mutation operator depends on the activation of the proximity sensors in various combinations (Fig. 3b), since, when activated, they allow the selected individual to be changed to a new one. In another case, one of the individual parameters is selected randomly and a random value is added between -0.1 and 0.1, as shown in the algorithm 1. This



(a) Fitness value measured with the ultrasound sensor

(b) Combinations of proximity sensors

Fig. 3: Sensors available in the EMeRGE robot

mutation operator is used in all the optimization techniques implemented in this work.

P: Population
a: Individual
$a \leftarrow$ Select individual(P);
if *Active proximity sensor* **then**
  $a \leftarrow$ New individual();
end
else
  M $\leftarrow$ Select parameter(a);
  S $\leftarrow$ Random $(-0.1, 0.1)$;
  M $\leftarrow$ M+S;
  $a \leftarrow$ Replace parameter(M);
end

Algorithm 1: Mutation function

Hill climbing and Simulated annealing: These optimization algorithms are used to solve optimization problems iteratively. In both cases, the best known individual is temporarily stored and used to generate a new one by applying a mutation. When applying the mutation, the individual generated is evaluated and each technique has an acceptance parameter, which are: in the case of hill climbing [22], the best known individual is replaced by the one generated if it is better. Similarly, the Simulated annealing technique accepts a new individual, if

this is better than the known one. In addition, it adds a condition of acceptance of a new individual, which depends on a temperature value determined by the equations 7, 8 and a random number [18] (F_1 =current individual, F_2 = Best individual).

$$\triangle F = F_1 - F_2 \tag{7}$$

$$P(\triangle f, \tau) = e^{-} \triangle f/\tau \tag{8}$$

Genetic Algorithm (GA): It is a population optimization technique, that is, optimizes sets of individuals to find a solution to a problem. In this work, a population of ten (10) individuals was optimized using the parameters described below, following the scheme proposed in algorithm 2.

1. *Initial population:* The way to generate the individuals is the same as mentioned above, each generated individual is temporarily stored in the centralized controller.
2. *Selection:* The selection mechanism used is based on the roulette method and its objective is to select the most suitable individuals to form the next generation. Initially the centralized controller evaluates each individual and then selects them.
3. *Cross:* The crossing of two individuals allows combining their characteristics to form similar ones and incorporate them into the population. In this work, a cross-over by combination of linear factors was implemented, which consists of adding and multiplying ordered pairs of the components of the selected individuals.
4. *Mutation:* The mutation of a randomly selected individual is performed in the manner described above.
5. *Generational replacement:* The generational replacement is carried out directly, that is, the population saved is replaced by the population to which the operations have been applied; selection, crossing and mutation.

```
d: distance traveled
P0 ← New popuation(10);
while Stop condition not active do
    d ← Evaluate(P0);
    P1 ← Select individuals(P0);
    if Random(0,1)<0.7 then
        P1 ← Cross-over(P1);
    end
    if Random(0,1)<0.1 then
        P1 ← Mutation(P1);
    end
    P0 ← P1;
end
```

Algorithm 2: Genetic algorithm structure implemented.

4 Experimental result

The different configuration parameters of the robot and the fitness value were stored automatically in a computer, using the CAN sniffer that was incorporated into the robot. The sniffer was linked to an application that allows to export the information in plain text format[1].

The first test performed on the robot consisted of; connect three modules of the structure in vertical orientation and modulate the parameters of the CPG using the three optimization techniques in an environment without obstacles. The results obtained are presented in a graph (Fig. 4) with three different lines, which show the behavior of the best individual found during 150 iterations of each technique. In the case of the genetic algorithm, it is the value of the best individual in each generation.

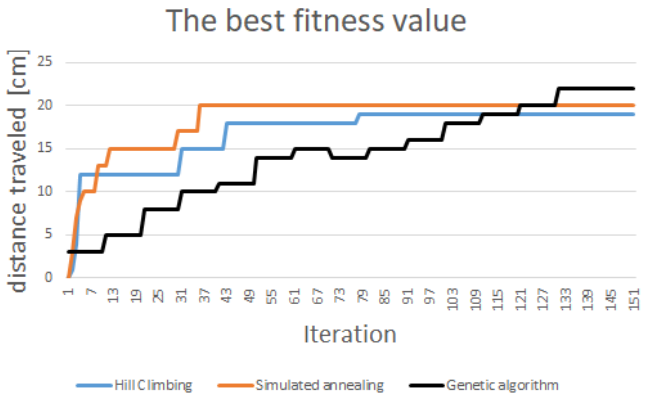


Fig. 4: Trend of the best value found using the three optimization techniques.

The second test performed consists of; propose two scenarios (Fig. 6) to determine if the robot adapts to different environments. On stage one (Fig. 6a), 150 fitness assessments were made and on stage two (Fig. 6b), 250 fitness assessments were made. The results obtained correspond to 5 executions of each optimization technique and are presented graphically in two box diagrams (Figs. 6c and 6d), which show the behavior of the best individuals found.

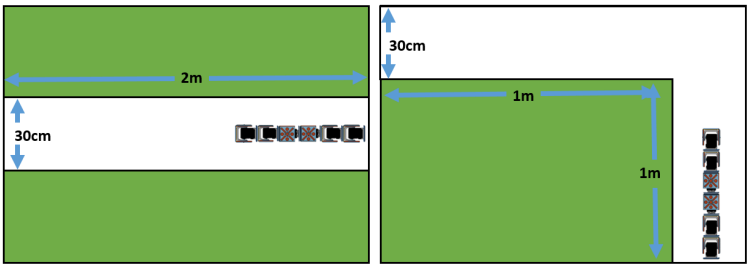
Finally, a genetic algorithm test was performed to determine if there were sets of movements that would allow a robot to evade an obstacle above. The staircase has two steps; The first step is 3 cm high and the second is 6 cm. To avoid this obstacle, 50 generations of the genetic algorithm were executed, as shown in the frames (Fig. 5).

[1] The graphical user interface, the programs and steps necessary to assemble the robot are available at the following link: https://sites.google.com/view/emergemodular

Fig. 5: Frames of the EMeRGE robot evading an obstacle in the form of a ladder.

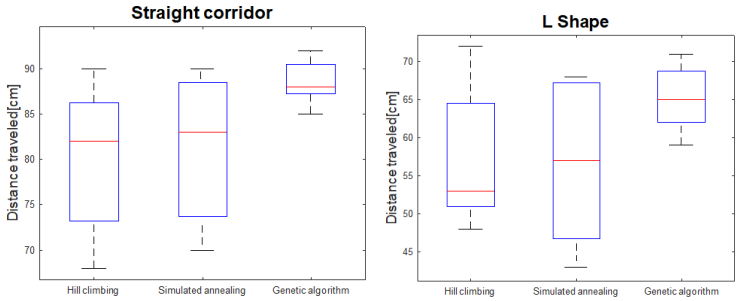
5 Discussion

The optimization techniques presented in this paper have been used extensively in solving different problems. Despite this, it can be said that the contribution made with this work consists of; the implementation of these optimization techniques in a centralized type controller, to optimize sets of parameters of the CPG from the sensory information, which allow the operation of a modular robot autonomously without the need for a previous simulation.



(a) The first scenario (straight corridor)

(b) The second scenario (L shape)



(c) Box plot (straight corridor)

(d) Box plot (L shape)

Fig. 6: Outline of the two proposed scenarios and the behavior of the best individuals found for each optimization technique.

Of the results presented as shown in the Fig. 4, the genetic algorithm showed a higher performance, since it found sets of movements that allow a robot with three modules to travel a distance of 22 centimeters. In addition, as shown in the box diagrams, GA is a technique that finds repeatable movements, that is, it allows the robot to generate the same movement schemes to solve a problem if it is on the right track. However, as observed in a segment of the executions made using the hill climbing algorithm (Fig. 7), There are sudden bursts of fitness that allow the robot to evade obstacles, this is because the fitness calculation is based the difference of two points and a high fitness value is assigned to a turning movement so that the robot does not move.

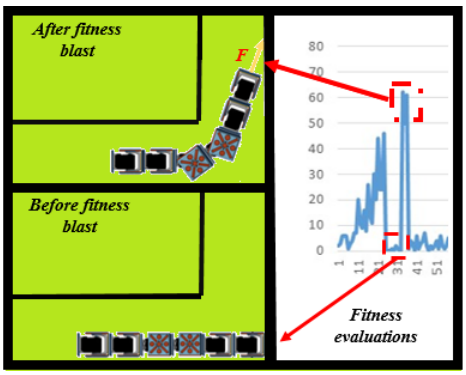


Fig. 7: Fitness explosion.

Future work is based on decentralized control strategies that allow the incorporation of optimization algorithms that work in parallel. That is, each robot can operate autonomously and synchronize with them forming a structure with other modules. However, a partial problem may arise because fitness is estimated locally. One way to improve the measurement of physical shape is to implement an artificial vision system that allows the robot to have a better perception of the environment. The idea is to maintain autonomy, therefore, a minicomputer will be implemented in the robot to avoid the incorporation of an external controller.

References

1. A. Brunete, M. Hernando, and E. Gambao. Offline ga-based optimization for heterogeneous modular multiconfigurable chained microrobots. *IEEE/ASME Transactions on Mechatronics*, 18(2):578–585, April 2013.
2. P. Chand. Fuzzy reactive control for wheeled mobile robots. In *2015 6th International Conference on Automation, Robotics and Applications (ICARA)*, pages 167–172, Feb 2015.
3. Avis H. Cohen, Philip J. Holmes, and Richard H. Rand. The nature of the coupling between segmental oscillators of the lamprey spinal generator for locomotion: A mathematical model. *Journal of Mathematical Biology*, 13(3):345–369, Jan 1982.

4. A. Crespi and A. J. Ijspeert. Online optimization of swimming and crawling in an amphibious snake robot. *IEEE Transactions on Robotics*, 24(1):75–87, Feb 2008.
5. Serge Kernbach, Eugen Meister, Florian Schlachter, Kristof Jebens, Marc Szymanski, Jens Liedke, Laneri Davide, Lutz Winkler, Thomas Schmickl, Ronald Thenius, Paolo Corradi, and Leonardo Ricotti. Symbiotic robot organisms: Replicator and symbrion projects, 01 2008.
6. D. Lachat, A. Crespi, and Auke Jan Ijspeert. Boxybot: a swimming and crawling fish robot controlled by a central pattern generator. In *The First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics, 2006. BioRob 2006.*, pages 643–648, Feb 2006.
7. G. Li, R. Urbina, H. Zhang, and J. G. Gomez. Concept design and simulation of a water proofing modular robot for amphibious locomotion. In *2017 International Conference on Advanced Mechatronic Systems (ICAMechS)*, pages 145–150, Dec 2017.
8. Liang Li, Chen Wang, and Guangming Xie. A general cpg network and its implementation on the microcontroller. *Neurocomputing*, 167:299 – 305, 2015.
9. C. Liu, J. Liu, R. Moreno, F. Veenstra, and A. Faina. The impact of module morphologies on modular robots. In *2017 18th International Conference on Advanced Robotics (ICAR)*, pages 237–243, July 2017.
10. D. Marbach and A. J. Ijspeert. Online optimization of modular robot locomotion. In *IEEE International Conference Mechatronics and Automation, 2005*, volume 1, pages 248–253 Vol. 1, July 2005.
11. J. Monsalve, J. Leon, and K. Melo. Modular snake robot oriented open simulation software. In *The 4th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent*, pages 546–550, June 2014.
12. Rodrigo Moreno, Ceyue Liu, Andres Faina, Henry Hernandez, and Jonatan Gomez. The emerge modular robot, an open platform for quick testing of evolved robot morphologies. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '17, pages 71–72, New York, NY, USA, 2017. ACM.
13. S. Murata and H. Kurokawa. Self-reconfigurable robots. *IEEE Robotics Automation Magazine*, 14(1):71–78, March 2007.
14. Pinhas Ben-Tzvi Paul Moubarak. Modular and reconfigurable mobile robotics. *ELSEVIER-Robotics and Autonomous Systems*, 1(60):1648–1663, 2012.
15. H. Suzuki, J. H. Lee, and S. Okamoto. Development of semi-passive biped walking robot embedded with cpg-based locomotion control. In *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 75–78, June 2017.
16. M. Tavakoli, C. Viegas, L. Marques, J. N. Pires, and A. T. de Almeida. Magnetic omnidirectional wheels for climbing robots. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 266–271, Nov 2013.
17. Y. Tian, V. Gomez, and S. Ma. Influence of two slam algorithms using serpentine locomotion in a featureless environment. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 182–187, Dec 2015.
18. P.J. van Laarhoven and E.H. Aarts. *Simulated Annealing: Theory and Applications*. Mathematics and Its Applications. Springer Netherlands, 1987.
19. V. Vonásek and J. Faigl. Evolution of multiple gaits for modular robots. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, Dec 2016.
20. V. Vonásek, S. Neumann, D. Oertel, and H. Wörn. Online motion planning for failure recovery of modular robotic systems. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1905–1910, May 2015.

21. W. Wu, Y. Guan, Y. Yang, and B. Dong. Multi-objective configuration optimization of assembly-level reconfigurable modular robots. In *2016 IEEE International Conference on Information and Automation (ICIA)*, pages 528–533, Aug 2016.
22. Bowei Xi, Zhen Liu, Mukund Raghavachari, Cathy H. Xia, and Li Zhang. A smart hill-climbing algorithm for application server configuration. In *Proceedings of the 13th International Conference on World Wide Web*, WWW '04, pages 287–296, New York, NY, USA, 2004. ACM.
23. M. Yim, W. m. 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*, 14(1):43–52, March 2007.
24. W. Zhao, Y. Hu, L. Zhang, and L. Wang. Design and cpg-based control of biomimetic robotic fish. *IET Control Theory Applications*, 3(3):281–293, March 2009.