Evolution of multiple gaits for modular robots

Vojtěch Vonásek
Dept. of Cybernetics, Faculty of Electrical Engineering,
Czech Technical University in Prague
Technicka 2, 166 27, Prague 6, Czech Republic

Jan Faigl
Dept. of Computer Science, Faculty of Electrical Engineering,
Czech Technical University in Prague
Technicka 2, 166 27, Prague 6, Czech Republic

***Abstract*—Modular robots are composed of many elementary mechatronic modules that can be connected to form a robot body of various shapes. This feature allows such a robot to adapt for a given task and particular environment. A motion of the modular robot is based on control of individual angles between the modules, and the robot locomotion can be realized using Central Pattern Generators (CPG). A robot motion in the environment with obstacles can be achieved using several locomotion controllers that are switched by a strategy based on motion planning techniques. Preparation of CPG-based gaits leads to a high-dimensional optimization that requires to design proper cost functions. Existing approaches optimize the gaits separately according to human-designed cost functions. In this paper, we investigate how to automatically derive a set of gaits suitable for modular robots without specifying low-level details about the gaits. We propose to optimize multiple gaits simultaneously using a single cost function. This cost function is based on the ability of motion planning to solve the task using the gaits being optimized. The proposed system is verified on several modular robots with unusual shapes including robots with failed modules.**

I. Introduction & motivation

Modular robots are composed of multiple building blocks which are connected using a docking mechanism [18], [30]. Contrary to other multi-DOF systems like humanoids, legged or snake-like robots, modular robots can be reconfigured to various shapes (examples are depicted in Fig. 1). Moreover, modular robots can recover from failures by ejecting or replacing broken modules.

Central Pattern Generators (CPG) [8] are oscillators providing rhythmic control signals for the individual actuators. To realize a gait, CPG parameters are optimized, e.g., using Genetic algorithms (GA). The design of a gait requires to decide what motion the gait has to perform, and to evaluate the motion using a suitable cost function. The gaits with corresponding cost functions are usually designed by human operators, that create them based on knowledge of a task to be solved and motion abilities of the robot. The specialized design of gaits is suitable in scenarios where human operators can anticipate behaviors of the robots. This is not always possible in modular robotics, especially for robots of unusual shapes or for robots with failed modules.

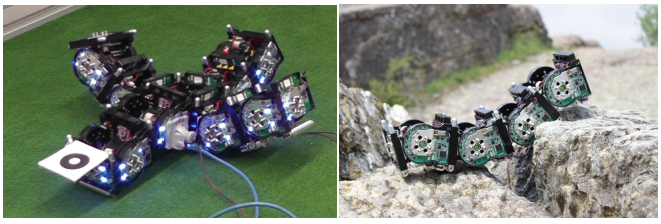


Fig. 1. Examples of modular robots made of CoSMO [14] modules.

CPGs are used mainly to generate basic rhythmic motions like 'crawl-forward' or 'swim', that can be optimized e.g., based on speed of motion [10]. Movements in environment with obstacles require various motion skills in order to avoid the obstacles. This can be achieved either using a single all-in-one controller [20], [5] or using multiple gaits that are switched by a high-level strategy [23], [27], [3]. The former approach is difficult for optimization, as the number of parameters is increased. Moreover, the evolved behavior (controller) needs to be evaluated repeatedly in the environment under various conditions, which is time consuming.

In the latter approach, a robot is equipped with several, relatively simple, gaits. The gaits are combined using motion planning in order to achieve a desired behavior. The motion plans are computed online considering the actual situation and replanned if the robot deviates from the planned trajectory. The optimization of the simple gaits is faster in comparison to optimization of single controllers realizing the whole task.

The crucial part of this approach still remains the design of proper cost functions for the individual gaits. In this paper, we investigate how to automatically derive a set of gaits for modular robots. Instead of a separate optimization of the gaits according to a set of predefined cost functions [29], we propose to optimize all gaits simultaneously. The point is to avoid specific knowledge of the gaits and cost functions and rather evolve all together. The only cost function, that is used to optimize the gaits, is based on the performance of motion planning with the gaits being evolved.

II. Related work

Two basic concepts are used to achieve motion of modular robots: self-reconfiguration and joint-control locomotion. In the concept of self-reconfiguration, the motion is achieved by repeated disconnection of the modules, moving them to a new position and reconnecting them back to the robot [6], [22], [16]. The joint-controlled locomotion, which is considered in this paper, is achieved by controlling angles of joints connecting the modules. This locomotion does not require reconfiguration and it can therefore be applied even for robots without self-reconfiguration capabilities.

Widely used approach to joint-control locomotion is based on Central Pattern Generators (CPG) [8]. CPGs are inspired by evidence from nature, that motion is generated by coupled neuro-oscillators providing rhythmic signals. CPGs have been used to generate locomotion of snake-like robots, to control swimming robots, to generate biped locomotion and also to control modular robots [10], [11], [2].

Behavior of CPGs is given by parameters like intrinsic frequencies or weights of coupling between the joints. Preparation of a gait leads to a high-dimensional optimization of these parameters, which is often solved using Genetic Algorithms [10], [11], [1]. The design of a suitable cost function is a crucial part of the GA-based optimization. The cost functions for a rhythmic locomotion can be based on speed of motion [10], [29], traveled distance [17] or consumed energy [24].

The design of cost functions for complex behaviors is however difficult [19], [4]. In the case of optimization using GA or other evolutionary methods, the cost function plays an important role and it is crucial for guiding of the evolutionary process. A suitable cost function has to provide enough selective pressure. This is not easy to achieve when optimizing behavior/controllers of robots, especially in the early stage of the evolutionary process. Individual solutions in the first iterations of GA usually do not posses significant ability to solve the problem and with an inappropriate cost function, the evolutionary process can turn to a blind random search [19].

This is the case of *aggregate* cost functions, that measure only a success ratio of performing a given task [15], [7]. As the initial population of controllers usually does not have any detectable competence to solve the task, the aggregate cost function cannot differentiate between the solutions. The behaviors are therefore evolved randomly and the possibility to complete the task by chance is small.

A possible solution is to employ several *incremental* cost functions [19]. First, a simple skill is optimized (e.g., a basic locomotion). Once the skill is achieved, a cost function for a more complex behavior (e.g., collision avoidance) is used. This process repeats until the desired behavior is achieved. The disadvantage of this approach is that the human operator still has to design the individual cost functions for each level of skills.

During the classic evolutionary-based optimization, various solutions are explored and evaluated using a given cost function. Many of these solutions are discarded due to their low performance in the desired task. However, some of these low-performing solutions can be useful for other tasks [3]. In [3], the potential solutions are ranked according to their quality and novelty in order to create a set of heterogeneous controllers for a walking robot. Each controller provides a gait in different direction. This approach, that is realized by a single run of optimization, results in a repertoire of various gaits that allows the robot to visit many places in the environment.

Complex behaviors can also be achieved using motion planning. Motion planning for many-DOF modular robots leads to a search in a high-dimensional configuration space. Sampling-based methods like Rapidly Exploring Random Tree (RRT) [13] can be used to explore the configuration space. The RRT method requires to generate suitable low-level control to move the robot in order to explore its configuration space. Recently, we have proposed a modification of RRT that employs CPG-based gaits for the exploration of the configuration space [29]. The gaits are considered as atomic actions on the motion planning level, which significantly speeds up the planning process. The resulting plans are sequences of motion primitives (gaits), and can be easily executed on robots. The advantage of this approach is, that the gaits can be realized by relatively simple locomotion generators. The global situation in the environment, like the presence of obstacles, is considered on the motion planning level that ensures that the robot avoid obstacles.

The crucial part of the motion planning with CPG-based gaits is the design of gaits together with the design of corresponding cost functions. In our previous works [29], [28], the cost functions were predefined for each gait by a specialist. In this paper, we investigate how to design the gaits automatically without providing the cost functions for the individual gaits. To achieve this goal, we propose to measure the quality of motion plans created from the gaits being evolved. This evaluation is similar to the aggregate cost functions [15], [7], [19]. To overcome the issues of the aggregate cost functions, we evaluate the quality of the created plans (their size, distance to goal) rather than only a success ratio. Motion planning can create feasible plans even with slow/inefficient gaits [26]. Consequently, these inefficient gaits can be evaluated by a non-zero cost function, which is important to to differentiate between the candidate solutions in the first iterations of GA-based optimization.

III. Single gait optimization

In this paper, we assume a chain-type modular robot [18] with m modules, where each module is equipped with a controllable joint. The number of actuators is therefore equal to the number of modules m. The joint-control locomotion requires to provide control signal $a_i(t)$, $i = 1, \ldots, m$ for each joint i. The signals $a_i(t)$ are generated using a CPG. The behavior of CPGs is determined by parameters that are collected in a parameter vector $\mathbf{x}^p$, where p denotes the gait.

For example, simple rhythmic motions can be realized with sine signals $a_i(t) = A_i \sin(\omega_i t + \varphi_i) + B_i$, where amplitude A_i, offset B_i, frequency ω_i and phase shift φ_i are the parameters. Such a gait p is then described by parameters $\mathbf{x}^p = (A_i, B_i, \omega_i, \varphi_i), i = 1, \ldots, m$ of the length $4m$. The duration of the gait p is τ^p.

For the optimization of gaits, we employ Particle Swarm Optimization (PSO) [12]. PSO is a population-based optimization method, where each particle represents a candidate solution $\mathbf{x}$. The quality of the solution is evaluated using a given cost function. Each particle has its position $\mathbf{x}_i$ (representing the solution) and velocity $\mathbf{v}_i$. Let $\bar{\mathbf{x}}_i$ denote the best solution of a particle i in its history and let $\hat{\mathbf{x}}$ denote the best solution among all particles. At the beginning, the

particles $\mathbf{x}_i$ are randomly placed into the search space and their velocities $\mathbf{v}_i(0)$ are set randomly. In each iteration, new velocity $\mathbf{v}_i(k+1)$ and position $\mathbf{x}_i(i+1)$ of the particle i is computed (in each dimension j) as:

$$\begin{aligned}\mathbf{v}_{i,j}(k{+}1) &= w\mathbf{v}_{i,j}(k)+\varphi_1 r_{1,j}(\bar{\mathbf{x}}_{i,j}-\mathbf{x}_{i,j})+\varphi_2 r_{2,j}(\hat{\mathbf{x}}_j-\mathbf{x}_{i,j})\\ \mathbf{x}_{i,j}(k{+}1) &= \mathbf{x}_{i,j}(k)+\mathbf{v}_{i,j}(k+1),\end{aligned} \quad (1)$$

where w, φ_1, φ_2 influence speed of the exploration of the search space and $r_{1,j}, r_{2,j}$ are random numbers from $U(0,1)$. The optimization terminates after a predefined number of iterations.

IV. Evolution of Multiple Gaits

Single gait is usually not suitable for complex environments, as robots need to move in various directions and avoid obstacles. This requires to equip the robots with multiple gaits that are switched by a motion planner. Instead of a separate optimization of the gaits [29], [28], we propose to optimize the gaits all at one time.

To optimize multiple gaits simultaneously, their parameters are collected into a single optimization vector $\mathbf{x}' = (\mathbf{x}^1, \ldots, \mathbf{x}^k)$, where k is the number of gaits to be optimized and $\mathbf{x}^i, i = 1, \ldots, k$ are the parameters of i-th CPG. This connection of parameter vectors $\mathbf{x}^i$ into a single vector is suitable for PSO, because the particle's best position $\bar{\mathbf{x}}_i$ and the global best position $\hat{\mathbf{x}}$ are combined with $\mathbf{x}_i$ separately in each dimension. It is therefore possible to simply concatenate the vectors x^i.

The task of the multiple-gait optimization is to find a vector $\mathbf{x}'$ which contains gaits that can be successfully combined by motion planning in order to solve the desired task. Here, the ability of the gaits contained in $\mathbf{x}'$ to solve the task is crucial. Therefore, the cost function for the vector $\mathbf{x}'$ is based on the performance of motion planning.

A. Motion planning with CPG-based gaits

The task of motion planning is to find a trajectory from a given start configuration $q_{init} \in \mathcal{C}_{free}$ to a desired goal configuration $q_{goal} \in \mathcal{C}_{free}$. A configuration of a modular robot with m joints $q = (x, y, z, \alpha, \beta, \gamma, a_i, \ldots, a_m)$ describes 3D position and rotation of a pivot module, a_i is an angle of the joint i. All possible configurations form the configuration space $\mathcal{C}$ and the free configurations, where the robot does not touch any obstacle, form the space $\mathcal{C}_{free} \subseteq \mathcal{C}$. The number of modules determines the dimension of $\mathcal{C}$, which is usually more than 10.

Motion planning in high-dimensional configuration spaces can be realized using sampling-based planners like Rapidly Exploring Random Tree (RRT) [13]. The main idea of RRT is to build a tree $\mathcal{T}$ of free configurations rooted at q_{init}. The tree is incrementally extended in each iteration until it approaches q_{goal} close enough.

The RRT-MP planner (RRT with Motion Primitives) is a modification of RRT that allows us to control the robot using CPG-based gaits [29]. In each iteration, a configuration q_{rand} is randomly generated in the configuration space and its nearest node q_{near} in the tree is found. The node q_{near} is then expanded using all gaits, which results in a set R of new configurations reachable from q_{near}. The expansion is realized using physical simulation, where the robot is placed at position q_{near} and controlled for a time τ^p by a gait p. During the motion, collisions with obstacles are checked. From the obtained set R of configurations reachable from q_{near}, the nearest one to q_{rand} is selected and added to the tree, but only if it does not lead to any collision during the movement. The algorithm terminates if the configuration tree approaches the goal configuration to a predefined distance or after a predefined number of iterations K_{max}.

To employ RRT-MP for the evaluation of the cost function for multiple gaits, we propose a modification of RRT–MP to create denser configuration trees. The algorithm is listed in Alg. 1. Instead of adding only a single node q_{near} in each iteration, the tree is extended by multiple configurations from the set R. To prevent an unnecessary filling of the tree using many nodes, the newly obtained configurations are pruned. A configuration $q_{new} \in R$ is added to the tree only if its closest distance to other node of the tree is higher than the distance to its parent q_{near} (line 9 in Alg. 1), which is depicted in Fig. 2b. This pruning technique ensures construction of dense trees [9].

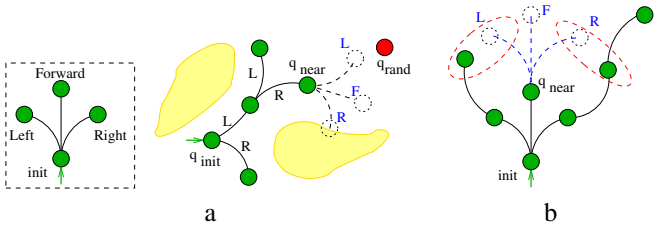


Fig. 2. Example of a configuration tree built by the RRT-MP planner for a robot equipped with three gaits. In the expansion step, these gaits are used to obtain new configurations reachable from q_{near} (dotted nodes) (a). These configurations are pruned if they approach other nodes in the tree closer than their parent node q_{near} (dashed ellipses). Only the configuration obtained with the 'F' gait will be added to the tree (b).

Algorithm 1: RRT-MP with flood-fill behavior

Input: initial configuration q_{init}, set of k motion primitives $\mathbf{x}^1, \ldots, \mathbf{x}^k$
Global params: number of planning iterations K_{max}
Output: configuration tree $\mathcal{T}$

1 $\mathcal{T}$.initialize(); // create empty configuration tree
2 $\mathcal{T}$.add(q_{init});
3 **for** $iteration = 1 : K_{max}$ **do**
4 q_{rand} = generate random configuration in $\mathcal{C}$;
5 q_{near} = nearest configuration to q_{rand} in tree $\mathcal{T}$;
6 **foreach** $\mathbf{x}^p \in (\mathbf{x}^1, \ldots, \mathbf{x}^k)$ **do**
7 q_{new} = apply CPG($\mathbf{x}^p$) to robot starting from q_{near};
8 q' = nearest neighbor to q_{new} in tree $\mathcal{T}$;
9 **if** $\varrho(q_{new}, q') > \varrho(q_{new}, q_{near})$ **then** // node pruning
10 $\mathcal{T}$.add(q_{new});
11 $\mathcal{T}$.addEdge(q_{near}, q_{new});
12 **return** $\mathcal{T}$; // constructed configuration tree

B. Cost function for multiple gaits

The cost function of the multiple gaits $\mathbf{x}'$ is computed based on the properties of a motion plan created from $\mathbf{x}'$. The main loop of the optimization process is listed in Alg. 2. We propose two basic methods to evaluate the cost function for the multiple gaits.

Coverage-based: In the scenario, where robots have to move between various places, the gaits should provide movements into various directions. The cost function for these gaits can be computed as the area reached by the motion plan, i.e., the area covered by the configuration tree. The environment is discretized to 2D or 3D cells (depending on the character of the environment), and the nodes of the tree are assigned to cells according to their 2D/3D positions. The cost function is measured as the percentage of covered cells and it has to be maximized. The gaits, that do not move robots at all, would result in zero cost function. However, as soon as the gaits (or one of them), move the robot, this cost function increases.

Distance-based: A different cost function is suitable in scenarios, where a robot is supposed to move only between particular places. In this case, the gaits should realize only motions required to visit the particular goal configuration. The distance-based cost function is computed as the distance between the motion plan constructed from the gaits being evolved and the desired goal configuration. In this case, the cost function has to be minimized. Measuring the cost function as the length of resulting path is also possible, but it is not suitable in the early stage of the optimization, where gaits are not sufficiently evolved. In such a case, motion planner cannot find path to the goal configuration and the cost function would have to be replaced by a different metric.

In both cases, tested gaits are used by RRT-MP to create motion plans. The maximum size of the motion plans (in the term of tree nodes) is determined by the number of planning iterations K_{max}. By setting K_{max} to a high number, RRT-MP can construct suitable plans even with slow gaits, that move the robots negligibly. The motion planner can therefore "amplify" the effect of the gaits. This is crucial in the first iterations of an evolutionary-based optimization, where most of the tested solutions tend to be slow a clumsy.

Algorithm 2: Optimization of multiple gaits

Input: k number of gaits to be evolved, p number of PSO particles, g number of PSO generations, $q_{init} \in \mathcal{C}_{free}$ initial state of the robot
Output: Evolved gaits $\mathbf{x}^1, \ldots, \mathbf{x}^k$

$\mathbf{x}'_1, \ldots, \mathbf{x}'_p$ = create population of p particles,where
    each particle is $\mathbf{x}'_i = (\mathbf{x}^1, \ldots, \mathbf{x}^k)$;
initialize particles $\mathbf{x}'_i$ randomly;
foreach *generation = 1:g* **do**
    foreach *particle i = 1:p* **do**
        $\mathcal{T}$ = motionPlanning(q_{init}, $\mathbf{x}'^1_i, \ldots, \mathbf{x}'^k_i$); //Alg. 1
        // evaluate cost function
        f_i = coverage-based($\mathcal{T}$) or distance-based($\mathcal{T}$);
    update best local position $\bar{\mathbf{x}}_i, i = 1, \ldots, p$;
    update global best position $\hat{\mathbf{x}}$;
    update position of particles according Eq. 1;
return $\hat{\mathbf{x}}$; // best solution is stored in $\hat{\mathbf{x}}$

V. Experimental verification

The proposed optimization of multiple gaits has been verified in simulation with CoSMO modular robots [14] using Sim simulator [25]. Each module is composed of two cubes connected by a 1-DOF joint. The size of each module is 1 mu (map unit). The experiments were performed with two typical shapes of modular robots (denoted as "Lizard" and "Quadropod") and two artificial shapes ("L-shape", "H-shape" and "T-shape"); the robots are depicted in Fig. 3.

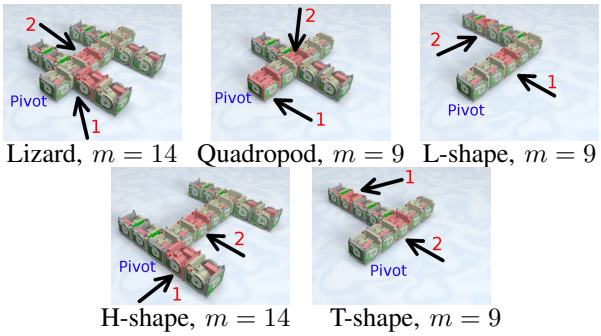
The locomotion is realized using Hopf-CPG [21]:

$$\begin{aligned} \dot{x} &= \alpha(\mu - r^2)x - \omega y \\ \dot{y} &= \beta(\mu - r^2)y + \omega x \\ \omega &= \frac{\omega_1}{e^{-by}+1} + \frac{\omega_2}{e^{by}+1}, \end{aligned} \tag{2}$$

where $r = \sqrt{x^2 + y^2}$ and ω is the frequency of oscillations in $\mathrm{rad.s}^{-1}$. The oscillator is controlled by parameters μ (amplitude of oscillations), ω_1 and ω_2 (frequencies of swing and stance phases, respectively) and constants $\alpha, \beta, b > 0$ that control the speed of the convergence to the limit cycle. The Hopf oscillator is used for each joint i. A gait p is therefore parametrized by vector $\mathbf{x}^p = (\mu_i, \alpha_i, \beta_i, \omega_{1,i}, \omega_{2,i}, b_i)$, $i = 1, \ldots, m$. The range of the variables suitable for the CoSMO robots are: $0.3 \leq \mu \leq 1.3$, $0 \leq \alpha, \beta \leq 50$, $0 \leq \omega_1, \omega_2 \leq 4$, $0 \leq b \leq 0.5$. The output from the Hopf oscillator is the variable x, which is the desired joint angle. The joint is controlled to this desired position by a PD controller. The duration of each gait is $\tau^p = 30$ s. The gaits have been optimized using PSO (Eq. 1) with 20 particles in 150 generations with the parameters $w = 0.1, \varphi_1 = \varphi_2 = 2$.

A. Optimization with coverage-based cost function

In the first experiment, the task is to optimize $k = 3$ gaits using the coverage-based cost function. The motion plans are constructed using RRT–MP (Alg. 1) in $K_{max} = 150$ iterations. The gaits are evolved in an environment without obstacles



Lizard, $m = 14$ Quadropod, $m = 9$ L-shape, $m = 9$

H-shape, $m = 14$ T-shape, $m = 9$

Fig. 3. Robot morphologies used in experiments. The arrows 1 and 2 denote the modules that are fixed in the experiments with failed modules. In the experiments C_1 and E_1, only the module denoted as 1 is fixed, while in the experiments C_2 and E_2, both modules are fixed.

(denoted as Empty) and with a single obstacle in the middle (denoted as Column) (Fig. 9). The size of both environments is 48×27 mu. For each environment and each robot, the optimization is run 20 times. The gaits are evolved for fully functional as well as partially damaged robots. One of typical failures of modules is the inability to control the hinge. In the experiment with damaged robots, one or two modules are stuck in a zero position and they cannot move. The damaged modules are highlighted in Fig. 3.

The results (average over 20 trials) are shown in Tab. I, where the first column denotes the environment (E–Empty, C–Column) and the subscript denotes the number of damaged modules (e.g., C_1 denotes robots with one failed module in the Column environment). The numbers in the first row of each environment show the achieved percentage coverage (mean and std. deviation).

All robots achieve higher coverage in the Empty environment than in the Column environment. The obstacle in the Column environment blocks the growth of the configuration tree, therefore the tree cannot cover the regions behind the obstacle which results in a lower coverage. Lower coverage is also achieved with broken robots (rows E_1, E_2, C_1 and C_2 in Tab. I). The robots with broken modules move slower as the failed modules decrease the number of controllable degrees of freedom. Due to slower motions, the gaits can drive the robot to smaller distance during the gait time τ and the constructed configuration trees cover less area.

The numbers in parentheses show the initial coverage achieved in the first iteration of PSO. The initial coverage is non-zero which indicates that the PSO optimization can be driven by the cost function even in the early stage of the optimization.

The coverage is influenced by the number of planning iterations that are used to construct the configuration tree. Higher number of planning iterations K_{max} brings advantage in the beginning of the evolution. In the first generations, the gaits are rather clumsy and they do not move the robot significantly, but even with such gaits, the planner can construct motion plans with nonzero coverage. However, too high number of planning iterations may be disadvantageous later, as the planner is always able to build a configuration tree that covers the area. A high number of planning iterations will result in slower gaits.

The resulting gaits are described by a parameter vector $\mathbf{x}'$, which can be translated to control signals using the employed CPG. In our case, the control signals are determined by the Hopf oscillator (Eq. 2). To visualize the gaits for many-DOF modular robots, we rather show how each gait moves the robot's pivot module, than show all control signals. Examples of the resulting gaits for Lizard robot are depicted in Fig. 4 and gaits for T-shape robot are depicted in Fig. 5. Two out of three evolved gaits are usually longer than the last one. The longer gaits are responsible for fast movements of the robot. In the motion planning, these fast gaits ensure that the configuration tree can rapidly explore the configuration space and consequently, cover the environment. An example of motion plan for Lizard is depicted in Fig. 6.

TABLE I
VALUES OF THE COVERAGE-BASED COST FUNCTION: MEAN/STD. DEVIATION. THE NUMBERS IN PARENTHESES SHOW THE COVERAGE IN THE INITIAL GENERATION OF PSO.

Map	Lizard	T-shape	L-shape	H-shape	Quadropod
E	32.2 / 1.3 (5.8 / 0.5)	42.5 / 6.0 (7.0 / 2.0)	52.1 / 5.2 (12.4 / 3.0)	27.6 / 3.0 (4.8 / 0.6)	28.8 / 2.9 (6.0 / 1.0)
E_1	24.6 / 4.0 (4.6 / 1.0)	40.0 / 3.5 (6.5 / 1.3)	49.6 / 5.0 (10.5 / 2.2)	20.0 / 2.3 (3.7 / 0.6)	22.5 / 1.8 (5.1 / 0.9)
E_2	21.7 / 2.8 (4.0 / 0.5)	34.1 / 3.5 (7.1 / 1.4)	48.3 / 4.2 (9.6 / 1.7)	20.3 / 3.2 (3.9 / 0.6)	23.4 / 3.2 (5.3 / 1.0)
C	25.7 / 3.1 (4.8 / 0.9)	37.2 / 4.2 (6.1 / 1.3)	47.6 / 4.3 (9.6 / 2.1)	16.4 / 0.0 (4.3 / 0.0)	26.0 / 2.2 (5.4 / 0.9)
C_1	20.4 / 3.9 (3.9 / 0.7)	34.9 / 3.4 (5.9 / 1.1)	45.0 / 4.8 (8.8 / 1.8)	15.3 / 2.1 (3.1 / 0.6)	20.3 / 1.9 (4.6 / 0.6)
C_2	18.3 / 1.6 (3.5 / 0.7)	31.6 / 4.5 (5.9 / 1.0)	44.6 / 4.8 (9.1 / 1.4)	16.5 / 2.8 (3.0 / 0.5)	21.0 / 2.1 (4.9 / 0.8)

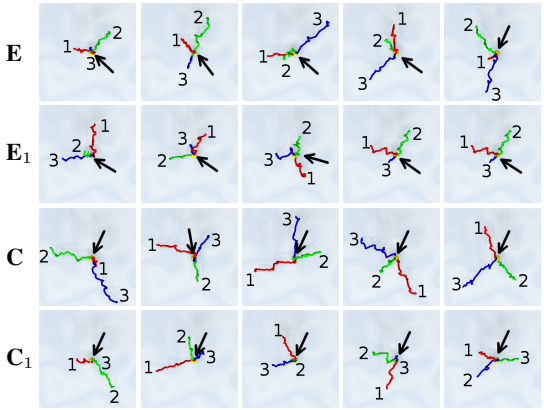


Fig. 4. Traces of the coverage-based gaits (top view) for the Lizard robot in the maps Empty and Column. The arrow denotes the initial position, the end points of the gaits are labeled 1, 2 and 3.

Each gait also realizes rotation of the robot, which is depicted in Fig. 7 for the Lizard robot. In this figure, the boxplots show the traveled distance and rotation of the gaits. In both graphs, the gaits are sorted in a descending order according to the traveled distance. In the Empty environment, the first (longest) gait of Lizard does not significantly rotate the robot (rotation is less than 3 degrees), while the remaining gaits cause higher rotation. In other cases, especially with broken modules, all gaits rotate the robot, which is caused by the failed modules.

B. Optimization with distance-based cost function

The distance-based cost function evaluates the ability of the planner to reach a given goal configuration. In this experiment, $k = 3$ gaits have to be evolved in order to reach the goal configuration q_{goal} placed 15 mu away from the initial position of the robot. The optimization is run 20 times for each robot and each environment. The cost function is evaluated as the distance between the constructed configuration tree and the goal configuration q_{goal}. The plans are created by RRT–MP with $K_{max} = 100$ iterations.

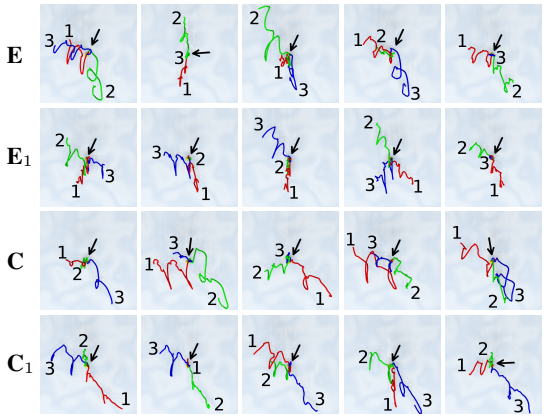


Fig. 5. Traces of gaits optimized using the coverage-based cost function (top view) for the T-shape robot in the maps Empty and Column. The arrow denotes the initial position, the end points of the gaits are labeled 1, 2 and 3.

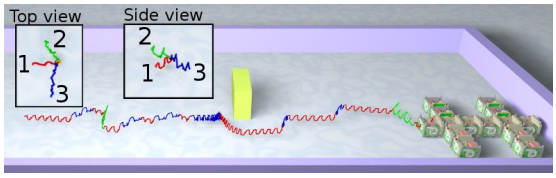


Fig. 6. Example of a motion plan for the Lizard robot in the Column environment. The plan is created from three coverage-based gaits.

The results are summarized in Tab. II, where the first line in each section is the distance to the goal configuration in the end of evolution (mean/std. deviation), and the second line shows the initial value of the cost function, i.e., this distance achieved in the first generation of PSO. During the optimization, the gaits allow the robot to approach the goal configuration to the distance less than 2 mu. The evolved gaits therefore enable the robot to traverse the distance 15 mu to the goal configuration.

Examples of resulting gaits are depicted in Fig. 8. In comparison to the gaits achieved using the coverage-based cost function (Fig. 4 and Fig. 5), the gaits optimized by the distance-based cost function are more straight and they point at the direction of the goal configuration. In the case of Lizard robot, one of the gaits is usually straight and long, while other two gaits are significantly shorter, which indicates that they move the robot negligibly. The long gait is however sufficient to reach to goal configuration. Similar results are achieved with the T-shape robot.

C. Utilization of gaits in other scenarios

The advantage of the motion planning with the gaits is that the gaits can be reused in other scenarios, which is verified in this experiment. The task is to find a trajectory between a center of the environment and a goal configuration placed in the distance 30 mu. The trajectory is planned using RRT–MP in $K_{max} = 100$ iterations. We aim to investigate the difference between the gaits optimized using the coverage-based and distance-based cost functions. As is described in the previous section, the distance-based gaits are oriented towards the goal used in the optimization. Therefore, three scenarios

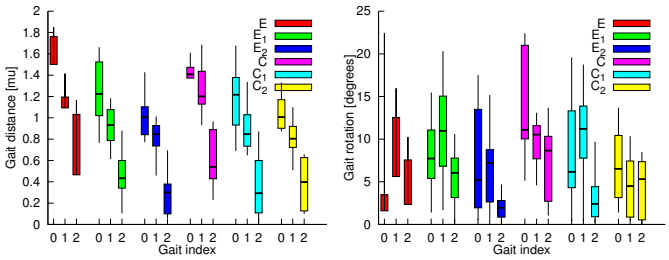


Fig. 7. The boxplots of the traveled distance (left) and rotation (right) of the evolved primitives for the Lizard robot. The primitives are sorted according their traveled distance. The boxplots are created from 20 measurements.

TABLE II
VALUES OF DISTANCE-BASED COST FUNCTIONS.

	Quadropod	T-shape	H-shape	Lizard	L-shape
E	1.31 / 0.32 (5.91/1.63)	1.14 / 0.38 (7.81/2.35)	1.45 / 0.29 (8.15/2.36)	1.33 / 0.34 (6.46/1.46)	0.97 / 0.39 (3.51/1.28)
C	0.93 / 0.37 (2.86/0.65)	0.52 / 0.30 (2.29/0.67)	0.85 / 0.42 (4.17/0.90)	1.28 / 0.25 (3.34/0.58)	0.59 / 0.31 (2.46/0.55)

are considered: 1) planning in the Empty environment in the same direction as was used for the optimization (Fig. 9a), 2) planning in the Column environment in the same direction as was used for the optimization, and 3) planning in the Column environment in the opposite direction, than was used for the optimization (Fig. 9b).

For each robot, 20 different triplets of distance-based gaits (oriented according to Fig. 9a) and 20 different triplets of coverage-based gaits are prepared. For each triplet, 20 motion plans are constructed, which results in 400 generated motion plans for each robot and scenario.

The average distance of motion plans to the goal configuration in these three scenarios are shown in Tab. III. All robots perform similarly. In the Empty environment (scenario 1), the goal is approached closer using gaits evolved with the distance-based cost function regardless the type of robot. This is expected behavior, as the distance-based gaits are oriented

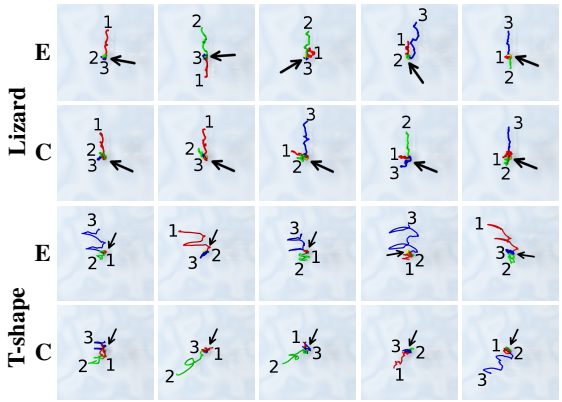
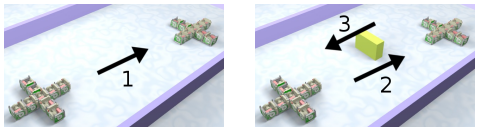


Fig. 8. Traces of gaits evolved using distance-based cost function. The arrow denotes the initial position, the end points of the gaits are labeled 1, 2 and 3.



(a) Empty (b) Column

Fig. 9. Direction of gaits achieved using the distance-based cost function (a). The arrows also show the direction of goal configuration in three scenarios (Section V-C).

in the same direction.

In the second scenario, better results are achieved using the coverage-based gaits. The coverage-based gaits provide motion to various directions, which is useful not only to achieved high coverage, but it is advantageous in the case of environments with obstacles. Contrary, the distance-based gaits rather provide one 'go-straight-ahead' primitive, which is not useful to avoid obstacles. The coverage-based primitives provide better performance also in the third scenario. In this case, the distance-based gaits are not useful, as they provide motion in the opposite direction and motion planning cannot combine them to find a plan.

Tab. IV shows the coverage of the constructed configuration trees. In all three scenarios and for all robots, a higher coverage is achieved with the coverage-based gaits. The influence of the coverage-based and distance-based cost functions to the motion plans is visualized in Fig. 10. The picture show the area that can be reached by a planner with $K_{max} = 100$ iterations. The highest coverage is achieved with healthy robots with coverage-based gaits optimized in Empty and Column environments. The lowest coverage is achieved with the distance-based gaits.

TABLE III
THE DISTANCE TO THE GOAL CONFIGURATION (IN MAP UNITS) FOR GAITS EVOLVED USING DISTANCE-BASED COST FUNCTION (COLUMNS *Dist.*) AND USING THE COVERAGE-BASED COST FUNCTION (COLUMNS *Cov.*). THE TABLE SHOWS AVERAGE/STD. DEVIATION.

Scenario	**1**		**2**		**3**	
Cost f.	**Dist.**	**Cov.**	**Dist.**	**Cov.**	**Dist.**	**Cov.**
L-shape	4.3/3.4	5.6/4.6	6.4/2.9	2.3/1.2	22.1/3.8	17.6/6.6
Quadro.	5.1/4.4	6.6/4.4	6.7/2.8	4.4/2.6	26.5/2.0	22.7/4.8
T-shape	7.7/4.4	10.0/4.7	6.4/2.3	3.2/1.5	23.2/4.7	24.6/5.4
H-shape	6.2/5.3	10.4/4.5	10.1/1.3	4.6/3.2	28.1/1.4	26.1/3.6
Lizard	4.1/4.2	8.3/3.8	8.7/2.5	5.8/2.9	27.3/1.5	24.2/3.4

D. Environment with stair

In the last experiment, the task is to find $k = 2$ gaits for traversing 5 stairs. Each stair is 0.6 mu high in order to allow the robot to enter them. The gaits are optimized using the distance-based cost function based on motion plans generated in $K_{max} = 150$ iterations. The goal is placed behind the stairs, so the cost function is minimized if the robot moves over the stairs. In this case, the gaits are found only for the Lizard and Quadropod robots. An example of a resulting trajectory over the stairs for the Lizard robot is depicted in Fig. 11a. The

TABLE IV
THE COVERAGE (IN PERCENTAGE) OF THE MOTION PLANS WITH GAITS EVOLVED USING THE DISTANCE-BASED COST FUNCTION (COLUMNS *Dist.*) AND USING THE COVERAGE-BASED COST FUNCTION (COLUMNS *Cov.*). THE TABLE SHOWS AVERAGE/STD. DEVIATION.

Scenario	**1**		**2**		**3**	
Cost f.	**Dist.**	**Cov.**	**Dist.**	**Cov.**	**Dist.**	**Cov.**
L-shape	7.7/3.0	31.6/12.2	6.9/2.7	21.1/7.4	6.4/2.9	18.3/5.4
Quadro.	5.4/1.7	16.1/6.3	4.9/1.5	12.8/5.5	3.4/1.0	10.1/3.8
T-shape	8.2/4.3	17.9/7.7	6.6/2.5	12.8/4.3	6.6/3.0	11.5/3.3
H-shape	4.5/1.7	12.8/4.2	3.4/1.2	8.5/3.4	2.6/1.0	7.7/2.5
Lizard	5.3/1.8	16.0/6.2	4.5/1.5	10.6/4.5	2.8/0.9	10.2/3.3

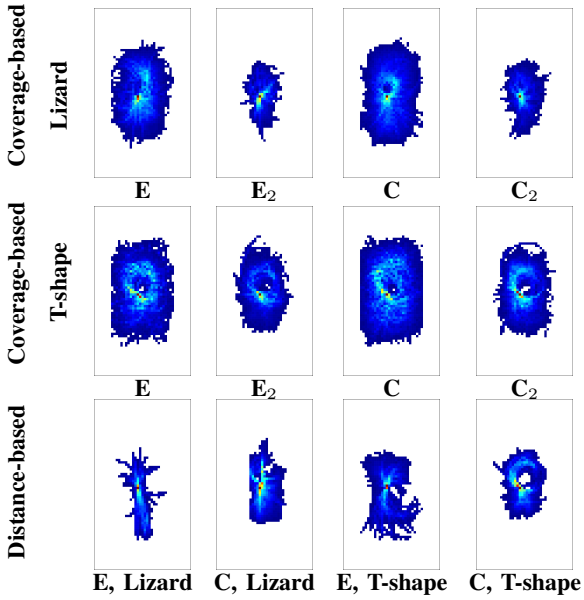
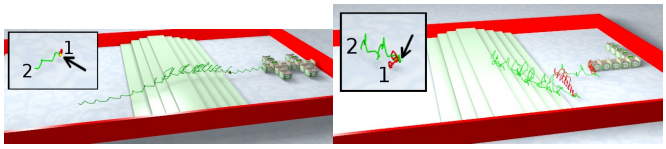


Fig. 10. Coverage achieved with gaits evolved according to the coverage cost function.

gaits for other robots are not optimized successfully because of too low number of planning iterations. During the allowed $K_{max} = 150$ iterations, other robots are not able to fully overcome the highest stair, which is shown in Fig. 11b. Due to this insufficient number of planning iterations, the gaits provide only motion on the plane and they are able to enter the first stair.

E. Discussion

The experiments have shown that the proposed system can derive multiple gaits using the coverage-based or distance-based cost functions. It is worth to mention that the gaits are evolved from scratch, i.e., the parameters $\mathbf{x}^i$ of individual



(a) Lizard (b) L-shape

Fig. 11. Example of motion plans in the environments with stairs.

gaits i are initialized randomly. Although no cost function is provided for the individual gaits, the resulting gaits provide reasonable movements. The experiments have been performed also with robots of unusual shapes (L- and T-shape robots). These robots cannot perform certain movements typical for robots with four legs, like the 'move-forward', as they always tend to rotate due to their shape. Suitable gaits have been evolved even for these robots, which is indicated by the achieved values of coverage-based as well as distance-based cost functions.

The comparison between the coverage-based and distance-based cost functions shows that the gaits resulted from coverage-based cost function are more universal and they provide motions into various directions. This allows the robots to move in large areas of the environment. In contrary, distance-based gaits are more straight and point in the direction of the given goal configuration. We refer to http://mrs.felk.cvut.cz/ssci2016 for more details and videos from the experiments.

VI. Conclusion

We presented a system for evolution of multiple gaits for modular robot. The proposed system employs Central Pattern Generators to realize the locomotion. Contrary to existing approaches, where all gaits have to be designed by a human expert, the proposed approach can derive all the gaits at once. In our approach, the quality of gaits is evaluated using a single cost function, that is based on the ability of motion planning to combine them in order to solve a desired task. Therefore, it is not required to anticipate motion abilities of the robots and their usefulness for a given task.

The system was verified on a set of modular robots including robots with partial damage. In both cases, suitable gaits were evolved only based on the qualities of the motion plans. The resulting gaits can be later reused in other scenarios, which is enabled by the motion planning.

VII. Acknowledgment

The presented work was supported by the Czech Science Foundation (GAČR) under research projects No. 15-09600Y and No. 16-24206S. Access to computing and storage facilities of the National Grid Infrastructure MetaCentrum provided under the programme (CESNET LM2015042) is greatly appreciated.

References

[1] S. L. Cardenas-Maciel, O. Castillo, and L. T. Aguilar. Generation of walking periodic motions for a biped robot via genetic algorithms. *Applied Soft Computing*, 11(8):5306–5314, December 2011.

[2] J. Conradt and P. Varshavskaya. Distributed central pattern generator control for a serpentine robot. In *International Conference on Artificial Neural Networks*, 2003.

[3] A. Cully and J.-B. Mouret. Evolving a behavioral repertoire for a walking robot. *Evolutionary Computation*, 24(1):59–88, March 2016.

[4] S. Doncieux and J.-B. Mouret. Beyond black-box optimization: a review of selective pressures for evolutionary robotics. *Evolutionary Intelligence*, 7(2):71–93, 2014.

[5] D. Filliat, J. Kodjabachian, and J. a. Meyer. Incremental evolution of neural controllers for navigation in a 6-legged robot. In *Proc. of the Fourth International Symposium on Artificial Life and Robotics. Oita*, pages 753–760. Univ. Press, 1999.

[6] R. Fitch and Z. Butler. Million module march: Scalable locomotion for large self-reconfiguring robots. *International Journal of Robotic Research*, 27(3–4):331–343, 2008.

[7] G. S Hornby, S. Takamura, O. Hanagata, M. Fujita, and J. Pollack. Evolution of controllers from a high-level simulator to a high dof robot. In *International Conference on Evolvable Systems*, pages 80–89. Springer, 2000.

[8] A. J. Ijspeert. Central pattern generators for locomotion control in animals and robots: A review. *Neural Networks*, 21(4):642–653, 2008.

[9] M. Kalisiak and M. van de Panne. RRT-blossom: RRT with a local flood-fill behavior. In *IEEE ICRA*, pages 1237 –1242, 2006.

[10] A. Kamimura, H. Kurokawa, E. Toshida, K. Tomita, S. Murata, and S. Kokaji. Automatic locomotion pattern generation for modular robots. In *IEEE ICRA*, 2003.

[11] A. Kamimura, H. Kurokawa, E. Yoshida, K. Tomita, S. Kokaji, and S. Murata. Distributed adaptive locomotion by a modular robotic system, M-TRAN II. In *IEEE IROS*, pages 2370–2377, 2004.

[12] J. Kennedy and R. Eberhart. Particle swarm optimization. In *IEEE International conference on Neural Networks*, pages 1942–1948, 1995.

[13] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning, 1998. TR 98-11.

[14] J. Liedke, Rene M., L. Winkler, and H. Woern. The collective self-reconfigurable modular organism (CoSMO). In *IEEE/ASME Inter. Conference on Advanced Intelligent Mechatronics*, pages 1–6, 2013.

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

[16] W. Liu and A. FT. Winfield. Distributed autonomous morphogenesis in a self-assembling robotic system. In *Morphogenetic Engineering*, pages 89–113. Springer, 2012.

[17] I. Macinnes and E. Di Paolo. Crawling out of the simulation: Evolving real robot morphologies using cheap reusable modules. In *Intl. Conf. on the Simulation and Synthesis of Life*, pages 94–99, 2004.

[18] P. Moubarak and P. Ben-Tzvi. Modular and reconfigurable mobile robotics. *Robotics and Autonomous Systems*, 60(12):1648–1663, 2012.

[19] A. L. Nelson, G. J. Barlow, and L. Doitsidis. Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems*, 57(4):345–370, 2009.

[20] F. Pasemann, U. Steinmetz, M. Hülse, and B. Lara. Evolving brain structures for robot control. In *International Work-Conference on Artificial Neural Networks*, pages 410–417. Springer, 2001.

[21] L. Righetti and A. J. Ijspeert. Pattern generators with sensory feedback for the control of quadruped locomotion. In *IEEE ICRA*, pages 819–824, 2008.

[22] D. Rus and M. Vona. Crystalline robots: Self-reconfiguration with compressible unit modules. *Autonomous Robots*, 10(1):107–124, 2001.

[23] W.-M. Shen, M. Krivokon, H. Chiu, J. Everist, M. Rubenstein, and J. Venkatesh. Multimode locomotion via SuperBot robots. In *IEEE ICRA*, pages 2552–2557, 2006.

[24] B. W. Verdaasdonk, H. F. J. M. Koopman, and F. C. T. van der Helm. Energy efficient walking with central pattern generators: from passive dynamic walking to biologically inspired control. *Biological Cybernetics*, 101(1):49–61, 2009.

[25] V. Vonásek, D. Fišer, K. Košnar, and L. Přeučil. A light-weight robot simulator for modular robotics. In *International Workshop on Modelling and Simulation for Autonomous Systems*, pages 206–216. Springer, 2014.

[26] V. Vonásek, K. Košnar, and L. Přeučil. Motion Planning of Self-reconfigurable Modular Robots Using Rapidly Exploring Random Trees. In *TAROS Conference*, pages 279–290. Springer, 2012.

[27] V. Vonásek, D. Oertel, S. Neumann, and H. Worn. Failure recovery for modular robot movements without reassembling modules. In *10th International Workshop on Robot Motion and Control (RoMoCo)*, pages 136–141, July 2015.

[28] V. Vonásek, O. Penc, K. Košnar, and L. Přeučil. Optimization of Motion Primitives for High-Level Motion Planning of Modular Robots. In *Mobile Service Robotics: CLAWAR 2014: 17th International Conference on Climbing and Walking Robots and the Support Technologies for Mobile Machines*, pages 109–116, Singapore, 2014. World Scientific.

[29] V. Vonásek, M. Saska, L. Winkler, and L. Přeučil. High-level motion planning for cpg-driven modular robots. *Robotics and Autonomous Systems*, 68(0):116 – 128, 2015.

[30] 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, 2007.