2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Synergizing Morphological Computation and Generative Design: Automatic Synthesis of Tendon-Driven Grippers

Kirill D. Zharkov[1], Mikhail E. Chaikovskii[1], Yefim V. Osipov[1], Rahaf Alshaowa[1], Ivan I. Borisov[1,2], and Sergey A. Kolyubin[1]

***Abstract*— The design process of robotic systems is a complex journey that involves multiple phases. Throughout this process, the aim is to tackle various criteria simultaneously, even though they often contradict each other. The ultimate goal is to uncover the optimal solution that resolves these conflicting factors. Within this paper we propose a design methodology to generate linkage mechanisms for robots with morphological computation. We use a graph grammar and a heuristic search algorithm to create robot mechanism graphs that are converted into simulation models for testing the design output. To verify the design methodology we have applied it to a relatively simple quasi-static problem of object grasping. Designing a fully actuated gripper may seem simple, but we found a way to automatically design an underactuated tendon-driven gripper that can grasp a wide range of objects. This is possible because of its structure, not because of sophisticated planning or learning. To test the applicability of the proposed method in real engineering practice, we used it to create physical prototypes. Simulation results together with results of testing of physical prototypes are given at the end of the paper. The framework is open source and the link to GitHub is given in the paper.**

I. Introduction

Designing robots is a multiphase process aimed at solving a multi-criteria optimization problem to find the best possible detailed design. When creating something new, a creator needs to think about shape, mechanics, materials, sensors, controllers, and more to make the design real. When the designing process is manual, it is difficult to prove the optimal solution numerically because it relies on the designer's experience and engineering intuition [1], [2].

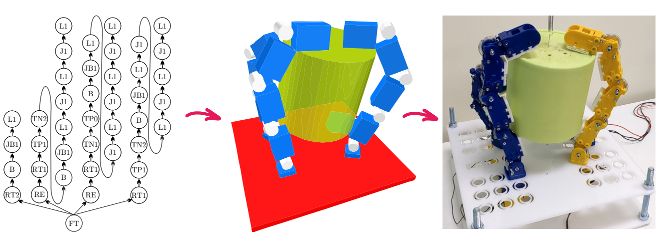

Fig. 1. The paper proposes a generative design approach that is based on interaction between a graph generated by a heuristic algorithm and a simulation model based on the graph. To verify generated designs and justify the proposed procedure, physical prototypes were built

*This work was supported by the Analytical Center for the Government of the Russian Federation (IGK 000000D730321P5Q0002), agreement No. 70-2021-00141.

[1]Kirill D. Zharkov, Mikhail E. Chaikovskii, Yefim V. Osipov, Rahaf Alshaowa, Ivan I. Borisov, and Sergey A. Kolyubin are with the Biomechatronics and Energy-Efficient Robotics Lab, ITMO University, Saint Petersburg, Russia `e-mail: {kdzharkov, s.kolyubin}@itmo.ru`

[2] Ivan I. Borisov is also with the SberRoboticsCenter, Moscow, Russia

A. Contribution

This research output is based on our previous studies on *morphological computation* as [3]–[5]. As a step towards, we developed the "rostok"[1] framework for generative and interactive design of linkage mechanisms. The proposed framework is general enough and an advanced user can use it for a linkage synthesis for any robotic purposes.

To verify applicability of the methodology for real engineering tasks, we have applied it to solve the task of computational design of underactuated tendon driven grippers, as a relatively simple quasi-static task. Fig. 1 shows steps of the design process: (1) a heuristic algorithm generates a graph; (2) the graph is converted into a simulation model; (3) virtual experiments are conducted, and the results are evaluated using a reward function; (4) the reward value obtained is used to guide the exploration of the design space; (5) the final design options are verified by means of physical prototypes.

For the sake of clarity, the contribution of the paper consists of several items:

- Combining *morphological computation* paradigm with *generative design*, the paper proposes an approach to the *automatic generation* of linkage mechanisms with "mechanical intelligence". The performance of these mechanisms is mainly determined by their *morphology*, rather than sophisticated trajectory planning and control.
- We successfully implemented the graph grammar approach to explore the design space of *anthropomorphic underactuated tendon-driven grippers*. Grasping is done by means of gripper morphology, rather than sophisticated control strategies.
- To ensure the highest performance of generated grippers, *physical prototypes* were constructed to validate both the solution and the overall methodology. This step aimed to assess the practicality and effectiveness of the methodology in real-world engineering applications.

B. Related Work

Model-based design optimization approaches can be used to efficiently synthesize linkage mechanisms together with needed control. Gradient-based together with global optimization algorithms are used to optimize geometry, mass distribution, or actuation within specific boundaries [6]–[9]. Properly formalized optimization tasks allow to efficiently find suitable solutions, however model-based design optimization approaches strongly depend on user inputs.

[1]`https://github.com/aimclub/rostok`

As an alternative approach, the generative design needs minimum input user to create unique and unexpected solutions. GD is intended to automatize co-optimization of mechanical structure and joints trajectories to satisfy poorly formalized tasks from a user. Generative adversarial networks (GANs) [10], graph neural networks (GNNs) [11], [12], and deep neural networks (DNNs) [13] are used to generate robots for a wide range of scenarios. A modular co-evolution strategy been implemented in [14] where primitive agents dynamically self-assemble into complex bodies and learn to coordinate their actions. The approach outperformed static methods in simulated environments, showcasing improved adaptability to environmental changes and agent structure variations. An innovative approach to robotic manipulator design, offering a streamlined pipeline that enables rapid creation and customization of manipulators with knitted, glove-like tactile sensors is given in [15]. By applying modular components and predefined rules, engineers can quickly prototype manipulator designs, fine-tune their shape, and seamlessly integrate tactile sensing, with successful real-world applications demonstrated. A novel underactuated compliant hand emulator, leveraging advanced simulation software and the adaptive synergy concept to simplify mechanical and control complexity, is introduced in [16].

II. Methods

In our approach the GD of dynamic systems can be decoupled into three sub-tasks: (1) *generation of the mechanism morphology*, (2) *optimisation of its kinematic and dynamics parameters*, and (3) *optimisation of control efforts*. Within this paper, the main focus is on the morphology of an underactuated tendon-driven griper, i.e. how many links and how they are jointed to form a mechanism, such that the gripper can grasp objects blindly without object-shape-based trajectory planning or behavior cloning. The only information regarding an object to be grasped is the position of its geometrical center expressed in gripper's palm frame.

A. Graph grammar

Our approach requires a suitable way to represent different structures and to use search algorithms. For the "rostok" we chose to encode mechanisms in the form of graphs. The graph representation is frequently used for robot design [11], [17]. In a graph, nodes typically symbolize components of mechanisms while the edges define their connections.

To effectively use the graph representation of designs, we must limit the search space to the graphs that represent mechanisms that can be manufactured, i.e., satisfy the conditions of existence. We apply the graph grammar for two reasons: limit the graph space and guide the search algorithm.

The graph grammar approach to graph synthesis requires two essential parts: *set of nodes* and *vocabulary of production rules* (Fig. 2). The set of nodes includes all possible node types that can appear in the graphs, however, several nodes of the same type can occur in a graph. Within the paper we limited a range of nodes to the minimal needed to generated planar open-chain linkages with revolute joints.

Production rules define possible graph transformations, mutating the current graph into a new one (Fig. 3). In this grammar scheme, the graph space is limited to graphs created by applying a sequence of rules. User can design the rule vocabulary and set of nodes in such a way that all graphs represent either the mechanism that can be assembled or some intermediate state that can be mutated to the mechanism representation via rule application. Therefore, production rules are the convenient way to look over the design space. The node alphabet and rules depend on the specific task of GD and the design space must be explored.

B. Reward function

We use the graph grammar to turn GD tasks into graph optimization problems. The optimization problem in graph space requires a mechanism of graph evaluation and comparison. This idea can be summarized into a *reward function* that can evaluate each proposed design. Through the reward function one tells the algorithm what is actually considered to be a good mechanism. The design of the reward function is the part of the GD permitting the vast space for variation and requires the most creativity.

In our paradigm of the GD, the algorithm has to *generate graphs* and *calculate reward* to evaluate the generated solution. Graph grammar approach is especially effective for building mechanism morphology, because it can restrict the graph space to only feasible mechanisms. In "rostok" we use graph grammar to search the morphology space. User can optimize robot parts by using global optimization algorithms or by incorporating it into the production rules. The same is true for control optimization tasks; they can be performed independently or added to the grammar rules, e.g. torques limits can be set directly to the graph of an actuated joints.

The "rostok" pipeline assumes the usage of the mechanism *simulations* as a main part of the reward calculation. Therefore, one needs to build an *automatic way to convert the graph into the simulation models* with the desired setup. In our experience, it is the hardest part of the process, since simulation models must satisfy *verification* and *validation* to minimize *reality gap*. The framework utilizes the PyChrono multiphysical modeling engine, the Python version of the Project Chrono C++ modeling library [18] for multi-body simulation of the design mechanisms.

As the reward calculation is simulation-based, our evaluation is limited to the graphs that depict the final robot design. However, it's important to note that the graph space encompasses the intermediate states as well. Any particular intermediate state can be used to isolate a graph subspace that can be approached from that state by applying the grammar rules. Therefore, one can use the finished mechanisms in the subspace for intermediate state evaluation.

C. Graph search algorithm

To explore the vast design space of possible solutions we need a heuristic search algorithm. The search space can be represented as a tree with all possible states. We can use any search algorithm to find solutions on the tree.

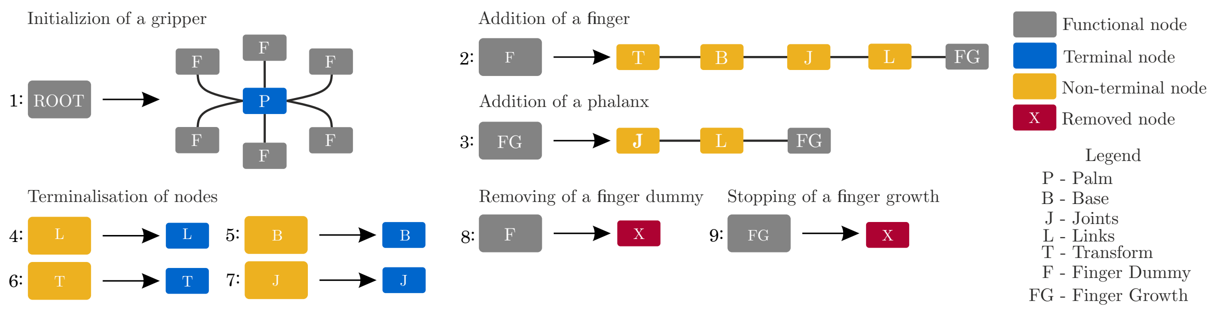

Fig. 2. The vocabulary of rules: rule 1 initializes a palm P with finger dummies F, rule 2 adds a finger, replacing a finger dummy F with a group of nodes, rule 3 adds a phalanx instead of node FG, rules 4-7 terminate nodes, rule 8 removes a finger dummy F, and rule 9 stops a finger from growing

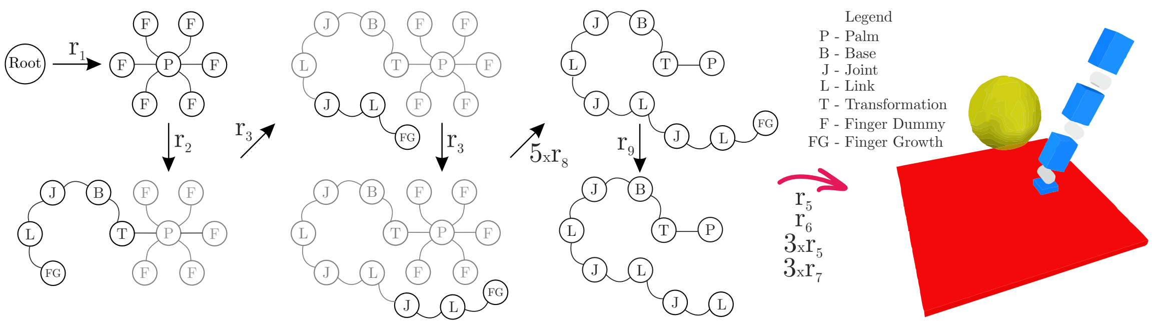

Fig. 3. Instructive example of a sequence of rules applied to generate the simplest one-fingered gripper: rule r_1 transforms an initial node into a palm with 6 finger dummies F; rule r_2 transforms one of dummy F's into a finger with a base B, joint J, and an attachment for the next link FG; rule r_3 adds extra phalanges, while rules r_8 and r_9 eliminate non-terminal nodes such as F and FG, finally rules r_5, r_6, and r_7 terminate non-terminal nodes

Monte-Carlo Tree Search (MCTS) [19] is well-suited for exploring the graph grammar-generated search space. The graph of the mechanism is a leaf in the search tree. The directed edges represent possible rules $\mathscr{R}^v$ from the rule dictionary $\mathscr{R}$. MCTS initiate the reward computation only for the leaf graphs in order to estimate the $\hat{V}$ value function of the states. The MCTS has only four steps: selection, expansion, simulation, and back propagation. During selection, the algorithm looks for a leaf node in the search tree. It applies the Upper Confidence Bound (UCB) method in each node until it reaches the leaf. In the expansion step, the algorithm applies valid rules $r \in \mathscr{R}^v$ to a current state. In the simulation step of the MCTS the estimation of child nodes rewards is occurred. The algorithm applies randomly selected rules until it reaches the terminal state. The special parameter restricts the application of rules, to guarantee that the algorithm always gets to the terminal state in finite number of steps. During backpropagation, the state-value function $Q(s,a)$ rules and node on the trajectory are updated.

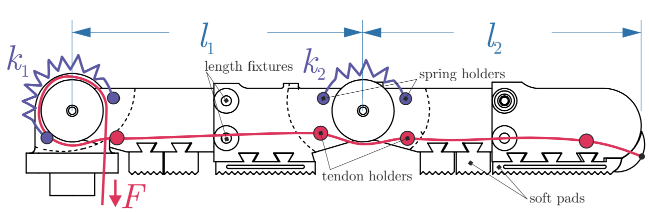

Fig. 4. Schematic representation of a finger with two phalanges in a horizontal state.

We implemented the MCTS algorithm because it is simple to implement, it showed acceptable results, and it can be used to compare other more efficient search algorithms, such as GNNs [11], that we plan to integrate later.

III. Generation of Tendon Driven Grippers

A. Finger design

We simplify the process of optimizing robot component parameters by selecting from a few specific values. The grippers we design consists of the palm and up to four fingers where each finger can have up to five phalanges. The palm serves as the central point of a star topology graph, with each finger branching out as a separate connection. This representation encompasses all mechanisms of a similar nature. Graph grammar rules consist of nodes and edges. The nodes contain all the information, while the edges determine the sequence of details. Thus, each finger consists of *phalanxes, joints* and *transforms* along the palm nodes. The length of each phalanx, stiffness of each joint and the position of a finger on a palm can be chosen from the predetermined discrete set, that must be given by a framework user.

For each finger (Fig. 4) algorithm discretely changes base's position and orientation on a palm, links' number and their lengths l_i, spring stiffness k_i, and driving force F. The upper and lower bounds for the parameters were manually selected based on the sizes of objects to be grasped and on-the-shelf components. We kept the pulley radius constant to simplify the procedure and focus on verifying the general method instead of finding an exact solution.

B. Production rules and robot components

The whole set of production rules is illustrated in Fig. 2. We use *terminal* and *non-terminal nodes* in our concept. Non-terminal nodes provide information about the robot's structure, while terminal nodes represent components with specific physical properties.

The terminal symbols on the scheme represent subsets of terminal nodes. These subsets are used to change the corresponding parameters. For example, the terminal L represents a set of discrete links' lengths, where each terminal node in the set has its own phalanx size. This graph grammar has two types of rules: (1) type one – changes design morphology, and (2) type two – transforms non-terminal nodes into terminal ones, determining the design parameters.

Even such a simple set allows us to generate vast space of the possible designs, which is another advantage of the graph grammar approach. The design space includes common designs like symmetric two and three finger grippers. It also allows for more unique variations, such as three fingers with different shifts and rotations on one side of the palm, and an opposing finger on the other side. Fig. 3 shows illustrative example of a sequence of rules that can be applied to generate the simplest one-fingered gripper.

C. Virtual experiment setup

The algorithm simulates the gripper's interaction with various objects at different torque values. Within this study we fixed the dimension of the objects range to three, thus, the evaluation of each mechanism includes $3 \cdot 3^n$ simulations. A simulation consists of two stages. At first stage the gripper attempts to grasp an object located in the center of the gripper workspace. To grasp objects blindly, we need to be able to hold them regardless of their initial orientation. To reduce its effect, we turn off gravity so that grasping is done only by using encompassing feature of grippers. At the second stage, if the grasp is successful, the algorithm applied an external load proportional to object weight. If the object remains grasped and does not fly away, then the grasp is considered successful.

D. Reward function

We designed several quantitative criteria to evaluate the performance of a mechanism in a simulation. Here we present a short description of each criterion.

1) Time period for the grasp: we abort the simulation prematurely if the mechanism cannot touch an object or the contact is lost. The simulation time can be used as a criterion:

$$r_1 = 1/(1+t_{sim}^2/t_{max}^2),$$

where $t_{sim} \in \mathbb{R}$ is the current time, $t_{max} \in \mathbb{R}$ is the limited simulation time. The criterion distinguishes unsuccessful designs, meaning that achieving longer contact with an object is preferable. Designs that do not touch the object receive zero reward. All mechanisms that successfully hold the object receive a score of 3 for r_1. For example, bad solutions depicted in Fig. 5, (a) have reward value less than 3 meaning that fingers only have touched the objects without grasping. That criterion is calculated when the object is grasped just before the external force to be applied.

2) The fraction of phalanxes contacting with objects: needed to reward for effective use of the phalanxes to prevent the growth of unnecessary components:

$$r_2 = \sum_{b \in \mathscr{B}} \mathbb{I}(b)/||\mathscr{B}||,$$

where $\mathscr{B} \in \mathbb{N}$ is a set of all mechanism bodies in simulation and $\mathbb{I}$ is an indicator function that equals to 1 if the corresponding body contacts an object; thus, the numerator of r_2 gets plus one for each body in contact with the object.

3) Dispersion of the contact forces: for a secure and stable grasp, phalanges have to apply similar forces to an object to be grasped

$$r_3 = 1/[1+\mathrm{std}(||f_i(t_{\mathrm{grasp}})||)],$$

where $f(t_{\mathrm{grasp}})$ is the force vector applied by a generated mechanism to a grasped object, std stands for standard deviation, t_{grasp} is time needed to secure the object.

4) Distance between an object center and a geometric center of contact points: the small distance reduce inertia effects while grasping and helps to achieve a stable grasp:

$$r_4 = 1/(1+||g_i - c_i||_2),$$

where $g_i \in \mathbb{R}^3$ is a geometric center of contact points and $c_i \in \mathbb{R}^3$ is a center of gravity.

5) Grasp time: time needed to secure the grasped object. Quick grasp gets higher reward:

$$r_5 = (t_{\max} - t_{\mathrm{grasp}})/t_{\max}.$$

6) The ability to withstand external force: when the grasp is secured, algorithm applies gradually increasing force to the grasped object. The force is proportional to the corresponding weight. Zero reward is given if the external force takes the object away from the gripper. If the gripper holds the grasped object, it gets the following reward:

$$r_6 = 1 - p(t_{\mathrm{final}})/p(t_{\mathrm{grasp}}),$$

where $p(t) \in \mathbb{R}^3$ is a position vector of a grasped object, t_{final} is the final time of the simulation.

7) Final reward: A generated gripper gets a scalar reward after each simulation, and we use its value to evaluate the design candidate:

$$r = w_1 r_1 + w_2 r_2 + w_3 r_3 + w_4 r_4 + w_5 r_5 + w_6 r_6,$$

where the weight coefficients w_i are equal to 1, except $w_1 = 3$, and $w_6 = 5$, because of higher priority. In Fig. 5, (a) samples are given. For example, a reward decomposition for the gripper with the best reward that holds a box:

$$r = 3 \cdot 1 + 2 \cdot 0.31 + 1 \cdot 0.36 + 1 \cdot 0.96 + 1 \cdot 0.77 + 5 \cdot 0.95 = 10.46.$$

We determine the highest value in the control space for each object we grasp. The sum of the highest rewards for all tested objects is then selected as the final reward for the design. Finally, the constructed reward function is used to search the design space applying the MCTS algorithm described in II-C.

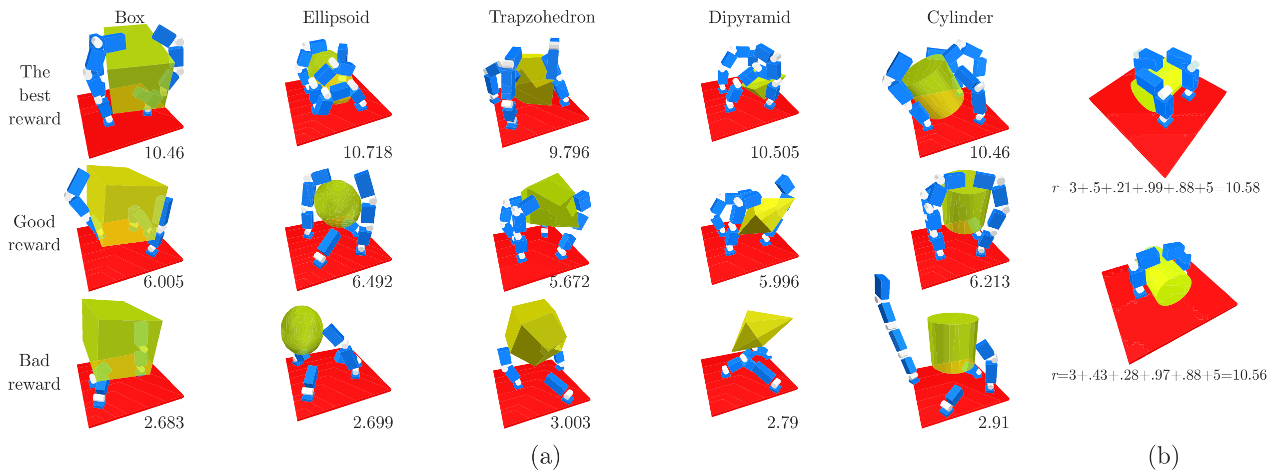

Fig. 5. Samples showing simulation models of generated gripper designs. Numbers next to images are values of a reward function used to evaluate the whole performance of generated graphs. Values around 10 indicate secure grasp of an object even with external force applied, values around 6 mean that an object was grasped, but was not able to handle an external force, and finally rewards around 3 means gripper have just touched an object without grasping.

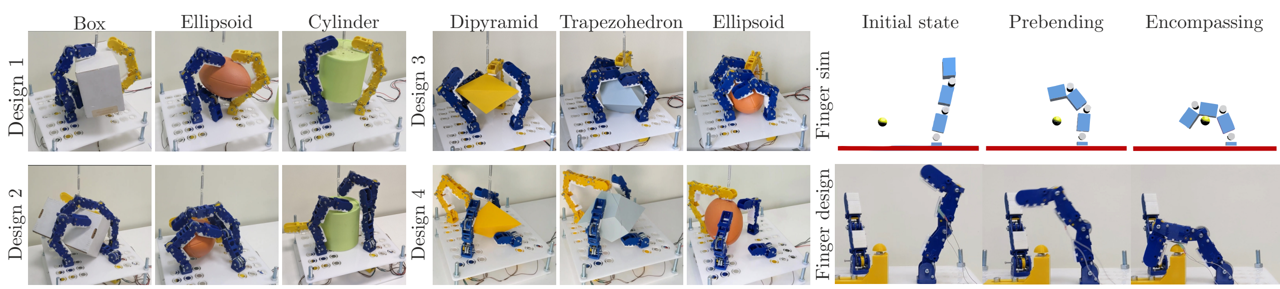

Fig. 6. Visualization of manufacturing physical grippers according to the generated designs with the top of reward function. Grasping is primarily done because of devices' morphology, rather than sophisticated control. Shown designs are capable of securely grasping different objects even with external force. Comparison between sequences of motion of a simulation model of one finger and corresponding physical prototype given on the right. To measure contact forces a penalty based method has been used in simulation, while for the physical prototype we have used a force resistive sensor

8) Remark on reward, bounds and initial state: The weight coefficients in the reward function have a substantial impact on the final designs, along with the initial orientation of objects and the limits set on the structure and parameters. If we limit the number of fingers to the conventional number of three together with limiting the number of phalanges we are getting generic grippers (Fig. 5, b). The main advantage of the proposed procedure is that the algorithm is capable of searching through the vast design space, and *what* will be the best solution depends on the reward.

IV. Results discussion

A. Search results

The algorithm evaluates and saves encoded mechanism representations and rewards during the design search process. We consider the designs with top rewards as a set of final results. The designs generated are shown in Fig. 5. The quantitative characteristics of the MCTS run are shown in Fig. 7. The increasing V and Q functions demonstrate that the search algorithm consistently achieves higher average rewards over time. The search space expands to include more valuable designs. A test run, that is performed in each step illustrates the current learned model of the search space.

After 40 iterations, the reward increases significantly. Using the learned design space model, the MCTS constructed a decision tree that extended to terminal states. It then proceeded to choose rules based on this tree. Once the learned MCTS has gone through a significant number of iterations, it constructs a decision tree that leads to terminal states with a highly promising outcome. Simulation took 2 days on a machine with CPU i9 10920X with 12 cores to conduct all calculations.

B. Experimental setup

We created a 3D printed physical setup to validate our approach and results (Fig. 6). The adjustable phalanx serves as the prototype's foundational block, featuring prismatic joints for length variation. This versatility allowed for various configurations of fingers, phalanxes, and lengths. The design includes tendon-driven features like pulleys and springs, mimicking virtual experiments for credible validation. A laser-cut palm component was designed to test all finger positions and orientations. Dynamixel AX-12A servos control the fingers due to their torque and feedback capabilities, while FSR402 sensors measure grasping pressure.

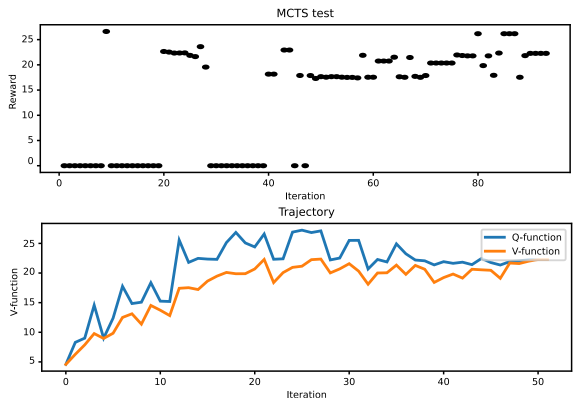

Fig. 7. Relation between reward values and a number of iterations together with plots for Q and V function for MCTS heuristic search algorithm

C. Verification

The tendon-driven finger is an underactuated open chain mechanism, thus its movement pattern and contact forces depend on both tendon tension and pulley positions. To verify the testing setup's precision, we conducted experiments with a single finger, illustrated in Fig. 6. We visually compared the movement of the real finger to the virtual one and used the FSR402 sensor to measure the differences in contact force. The difference between simulated and experimental contact forces did not exceed 34%.

It was noticed, that a larger number of fingers have been generated on the side where the external force has been applied. The grasp is more secured if the fingers wrap around from different directions like tentacles. Thus, as for the grippers on Fig. 6 we have analyzed the grasping patters visually. Because of underactuation, a significant effect on object initial state, and "flying" objects the motion of the fingers differs from the simulation. Nevertheless, because we have trained them on a set of objects with randomly applied forces, the grippers are versatile enough to encompass the objects even if they are oriented differently.

V. Conclusion

This paper explores the intricate process of designing robots, emphasizing the convergence of hardware and software in complex, conflicting criteria. We introduced a novel approach combining *morphological computation*, according to which the major portion of robots' desired behavior can be achieved with the "body" instead of the "brain", and *generative design*, highlighting the potential of automatic design for underactuated tendon-driven grippers, with open-source framework details and comprehensive testing results. The result of the "rostok"[2] pipeline for the task of generating tendon driven grippers is the set of designs with top rewards. The designs were thoroughly tested in a physical setup to ensure their ability to accurately reproduce the simulated kinematics and securely hold the object in place.

[2]All methods and algorithms described in the paper are available as a part of the open-source framework *rostok* (`https://github.com/aimclub/rostok`)

References

[1] Y. Sun, J. Falco, M. A. Roa, and B. Calli, "Research challenges and progress in robotic grasping and manipulation competitions," *IEEE robotics and automation letters*, vol. 7, no. 2, pp. 874–881, 2021.

[2] J. Pinskier and D. Howard, "From bioinspiration to computer generation: Developments in autonomous soft robot design," *Advanced Intelligent Systems*, vol. 4, no. 1, p. 2100086, 2022.

[3] I. I. Borisov, E. E. Khomutov, S. A. Kolyubin, and S. Stramigioli, "Computational design of reconfigurable underactuated linkages for adaptive grippers," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 6117–6123.

[4] I. I. Borisov, E. E. Khornutov, D. V. Ivolga, N. A. Molchanov, I. A. Maksimov, and S. A. Kolyubin, "Reconfigurable underactuated adaptive gripper designed by morphological computation," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 1130–1136.

[5] K. V. Nasonov, D. V. Ivolga, I. I. Borisov, and S. A. Kolyubin, "Computational design of closed-chain linkages: Hopping robot driven by morphological computation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 7419–7425.

[6] B. Thomaszewski, S. Coros, D. Gauge, V. Megaro, E. Grinspun, and M. Gross, "Computational design of linkage-based characters," *ACM Transactions on Graphics (TOG)*, vol. 33, no. 4, pp. 1–9, 2014.

[7] F. De Vincenti, D. Kang, and S. Coros, "Control-aware design optimization for bio-inspired quadruped robots," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1354–1361.

[8] T. Chen, L. Wang, M. Haas-Heger, and M. Ciocarlie, "Underactuation design for tendon-driven hands via optimization of mechanically realizable manifolds in posture and torque spaces," *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 708–723, 2020.

[9] G. Maloisel, C. Schumacher, E. Knoop, R. Grandia, and M. Bächer, "Optimal design of robotic character kinematics," *ACM Transactions on Graphics (TOG)*, vol. 42, no. 6, pp. 1–15, 2023.

[10] J. Hu, J. Whitman, M. Travers, and H. Choset, "Modular robot design optimization with generative adversarial networks," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 4282–4288.

[11] A. Zhao, J. Xu, M. Konaković-Luković, J. Hughes, A. Spielberg, D. Rus, and W. Matusik, "Robogrammar: graph grammar for terrain-optimized robot design," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–16, 2020.

[12] T. Wang, Y. Zhou, S. Fidler, and J. Ba, "Neural graph evolution: Towards efficient automatic robot design," *arXiv preprint arXiv:1906.05370*, 2019.

[13] A. Gupta, L. Fan, S. Ganguli, and L. Fei-Fei, "Metamorph: Learning universal controllers with transformers," *arXiv preprint arXiv:2203.11931*, 2022.

[14] D. Pathak, C. Lu, T. Darrell, P. Isola, and A. A. Efros, "Learning to control self- assembling morphologies: A study of generalization via modularity," in *NeurIPS*, 2019.

[15] L. Zlokapa, Y. Luo, J. Xu, M. Foshey, K. Wu, P. Agrawal, and W. Matusik, "An integrated design pipeline for tactile sensing robotic manipulators," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 3136–3142.

[16] A. Rocchi and K. Hauser, "A generic simulator for underactuated compliant hands," in *2016 IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*. IEEE, 2016, pp. 37–42.

[17] T. Wang, Y. Zhou, S. Fidler, and J. Ba, "Neural graph evolution: Automatic robot design," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=BkgWHnR5tm

[18] A. Tasora, R. Serban, H. Mazhar, A. Pazouki, D. Melanz, J. Fleischmann, M. Taylor, H. Sugiyama, and D. Negrut, "Chrono: An open source multi-physics dynamics engine," in *High Performance Computing in Science and Engineering: Second International Conference, HPCSE 2015, Soláň, Czech Republic, May 25-28, 2015, Revised Selected Papers 2*. Springer, 2016, pp. 19–49.

[19] L. Kocsis and C. Szepesvári, "Bandit based monte-carlo planning," in *Machine Learning: ECML 2006: 17th European Conference on Machine Learning Berlin, Germany, September 18-22, 2006 Proceedings 17*. Springer, 2006, pp. 282–293.