Evolutionary design of tall structures

Maciej Komosinski

Technical Report RA–06/12

Poznan University of Technology
Institute of Computing Science

maciej.komosinski@cs.put.poznan.pl

Abstract

This report presents results of optimization of three-dimensional designs in two tasks using various genetic encodings. Both tasks concern maximization of height; the goal is either maximization of the vertical position of the top vertex or maximization of the vertical position of the center of mass. Parameters of the simulation and optimization are covered, including discussion on the influence of non-deterministic noise and on the stopping criterion that is based on the number of non-improving evaluations. Eight genetic representations are described and their performance is demonstrated both in terms of fitness and the number of evaluations during evolution. Best designs resulting from optimization are shown as well.

Contents

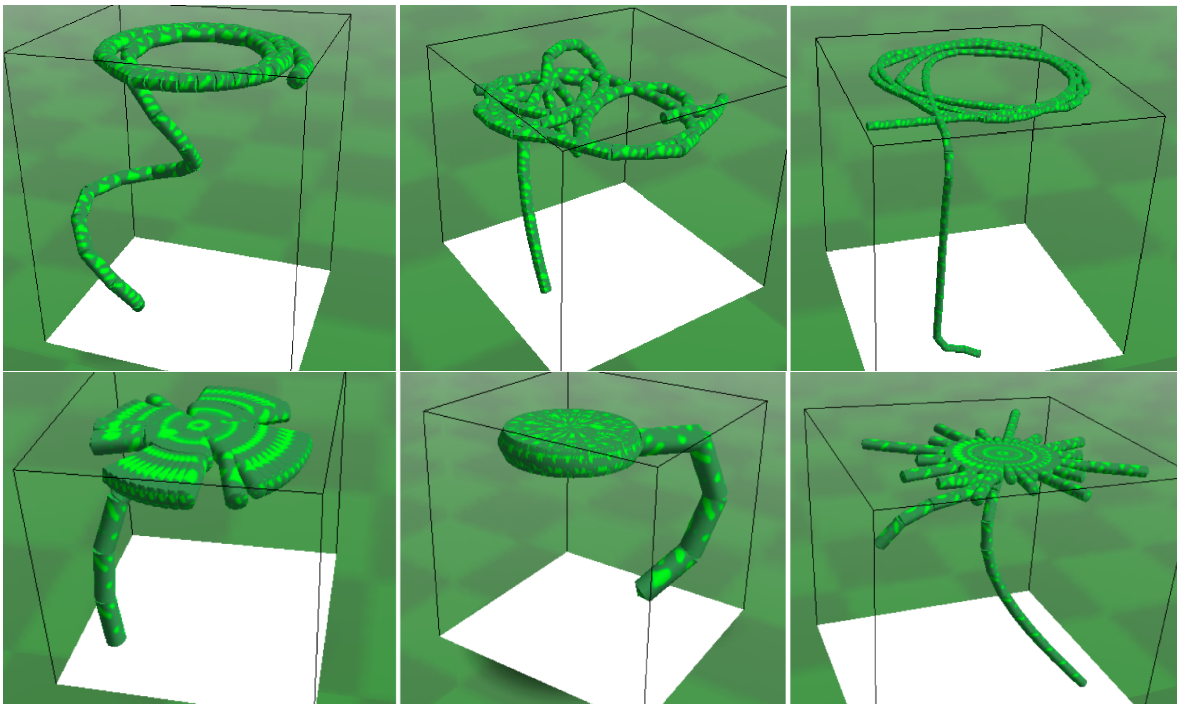

Figure 1: Optimization of tables demonstrates that multiple objectives must be usually considered in the area of evolutionary design.

1 Introduction

Designing structures has always been a domain of human experts; it traditionally required professional knowledge and experience. With the advent of computers came the possibility of simulation and evaluation performed *in silico*, and this, combined with optimization algorithms, allowed to run the process of design in virtual environments. Obviously, when design takes place within such an automated procedure, care must be taken to ensure that the simulation is accurate enough to reflect real behavior, including any non-deterministic factors and environmental noise.

From the perspective of optimization, the task of finding "best" designs is hard; the set of feasible solutions (designs) is usually infinite since the search space has a discrete-continuous character. Solutions contain variable amounts of information, there are strong dependencies between parts of a solution, there exist multiple local optima and numerous constraints. The evaluation of designs itself is difficult, being non-deterministic, time-consuming, multi-objective (Fig. 1), and, typically, delayed (the fitness of a construct is only known once the simulation has been completed). The optimization problem gets even more complex when active constructs are considered – i.e., constructs that are equipped with control systems coupled with their physical structure.

For these reasons, the key aspect in optimization is the representation of designs and reconfiguration operators that allow to change the designs. In the context of evolutionary optimization, the representation and the operators are called genetic. Their importance comes from numerous roles they play; artificial genetics brings structure to the search space, provides "building blocks", introduces biases, can reduce the solution space, and can be further characterized in terms of scalability, robustness, redundancy, compression, easiness of interpretation, support for modularity, etc. Genetic representations can also vary in terms of their interpretation (i.e., understanding the process of transformation from a genotype to a phenotype) – the mapping between genes and phenes may be direct or indirect, implicit or explicit, and the same genes may perform identical or different roles depending on the context or on the stage of development of a construct [22]. Genetics being a pivotal aspect in

optimization of designs, it is the subject of this work.

The computational experiments reported here concerned optimization of tall three-dimensional constructs; while these structures were evaluated in simulation, this setup is a prototype for real-world applications. Related research on evolutionary optimization in structural design and evolutionary design of tall constructs can be found in [16, 17, 29, 15, 3].

1.1 Simulation model

While genetic representation and its operators are important from the viewpoint of the optimization algorithm, the role of the representation itself is to describe a design (*any* design) that is considered feasible. This obviously depends on the experiment and the goal; various models of designs have been devised so far [31, 4, 5, 7, 30, 26, 9, 34]. Such models are usually created ad hoc – one model for one experiment or application, thus there is no reuse and no meaningful comparisons can be done. However, some kind of standardization is certainly possible. When we look at our surroundings, we will notice that many constructs and life forms can be modeled using graphs, with vertices and edges, as shown in Fig. 2.

The elements of the model that is considered in this work concern body (material structure) and brain (control system). The body is made of vertices ("Parts") and edges ("Joints"), and the brain is made of neural units (including signal processing, sensors, and actuators) and their connections. Control units can be embodied (fixed in body, i.e., assigned to Parts or Joints) or not.

In the model, Parts of the body are characterized by 3D position, orientation, physical properties like mass, friction, etc., and other properties that depend on the experiment (ingestion ability, color, etc.). Joints reference two Parts; their physical properties include axial and rotational stiffness, and other properties can include stamina, assimilation ability, etc. There are some constraints imposed on the body: at most one Joint can directly connect two Parts, each Joint must be connected with two distinct Parts, and all Parts must be directly or indirectly connected with each other. Brain is modeled as a network of any topology that contains "neural" control units with weighted connections. Each control unit can optionally have a list of properties (parameters). The model can represent a number of real constructs, including the ones popular in robotics [26, 28, 12]), architecture, design and engineering [8, 11, 33] (trusses, arches, bridges, scaffoldings, tensegrities [14], ...), biology [5], chemistry, and computer graphics [6], as illustrated in Figs. 2 and 3.

2 Genetic representations

To date, many genetic encodings have been proposed that are capable of representing 3D designs [33, 5, 28, 32, 1, 20, 11, 27]. They usually concern different simulation models, so their performance cannot be compared. This report takes the approach similar to the previous work of the author [20], where genetic representations constitute a hierarchy and each encoding must ultimately convert into a Direct encoding which describes a Model that can be simulated, as illustrated in Fig. 4.

The genetic encodings, apart from the conversion process, may additionally supply mapping information regarding individual parts of a genotype (genes), which may help in "genetic debugging" or tracing genes in evolution [22, 23]. Moreover, there may be more than one conversion path for each encoding as shown in Fig. 5.

It is important to realize that each encoding usually provides their own, specialized reconfiguration operators, so the optimization search takes place within the search space induced by the encoding. Genetic representations may differ substantially and may implement entirely different concepts (Fig. 6), so the topology and size of the search space is distinct for each encoding.

Using SDK [21], developers and researchers can easily add new representations and their operators to extend the hierarchy. Basic operators are mutation and crossover, and additionally, validity check, repair, and estimation of genetic similarity can be provided. Genetic encodings have unique numbers (*f0*, *f1*, etc.) and names as Fig. 7 shows.

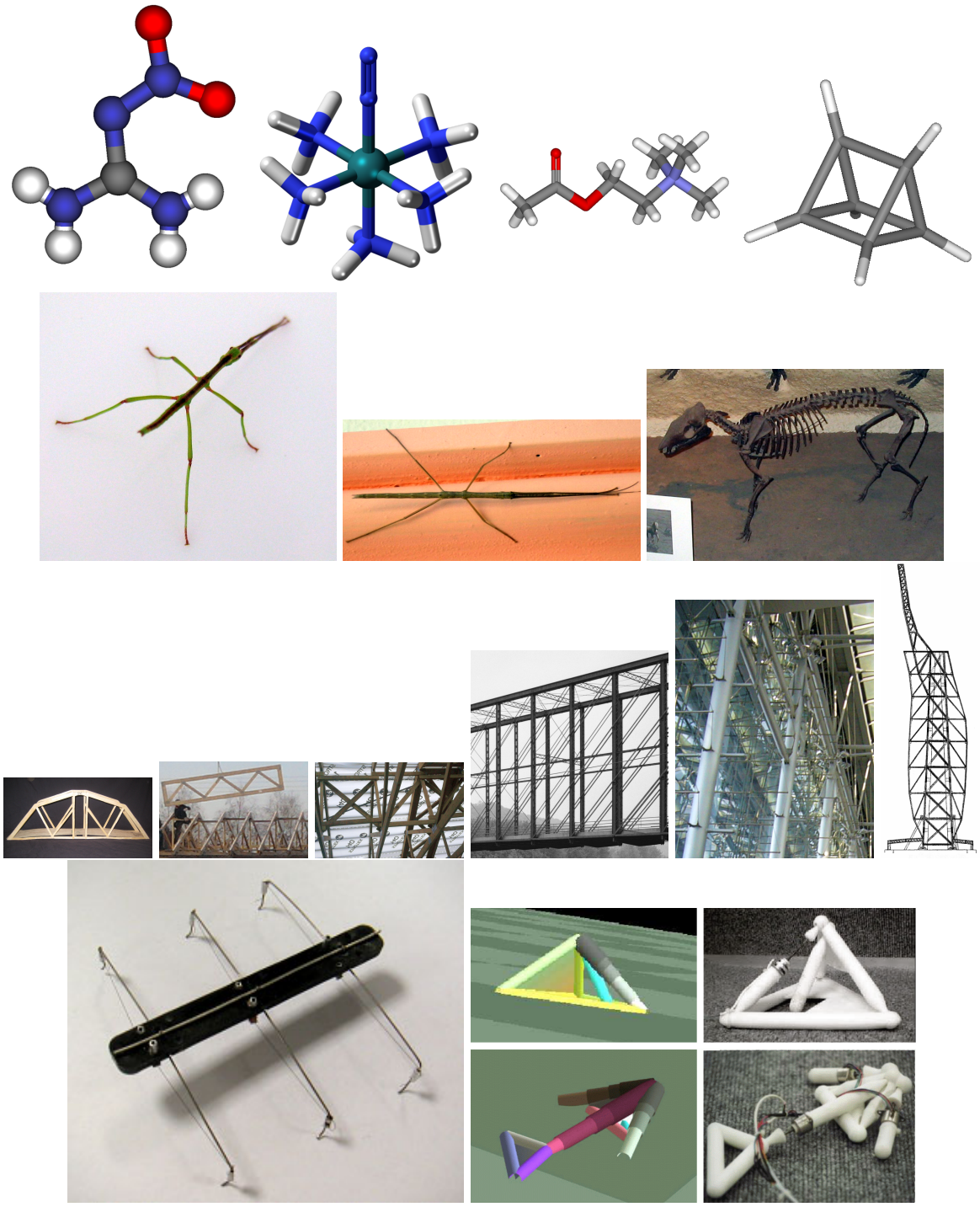

Figure 2: Examples of 3D constructs and 3D life forms that share a similar character: chemistry ("ball-and-stick" models), biology (stick insects – Phasmatodea and an Orohippus skeleton), engineering and design (including the "skeleton" truss tower of the Statue of Liberty on the right), and robotics (the Stiquito robot [2] and evolved stick robots [26]).

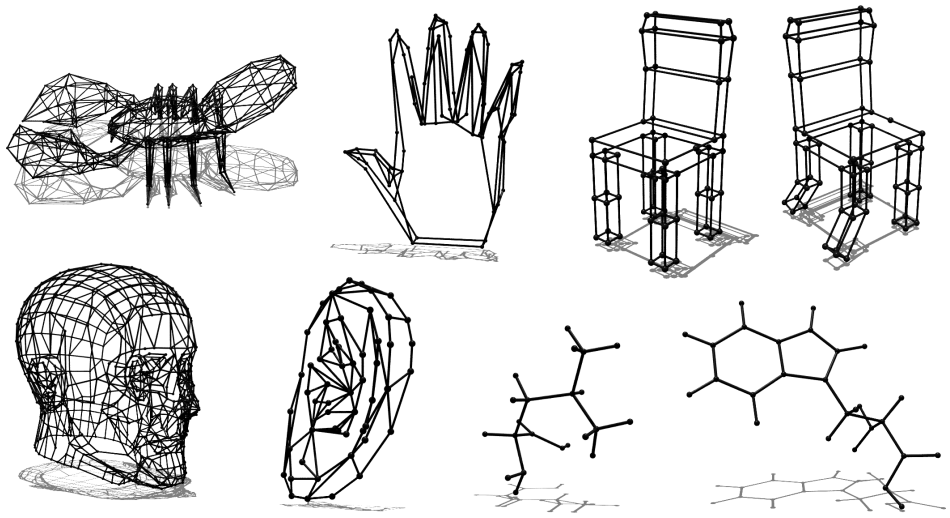

Figure 3: Three-dimensional graph-based models of real objects.

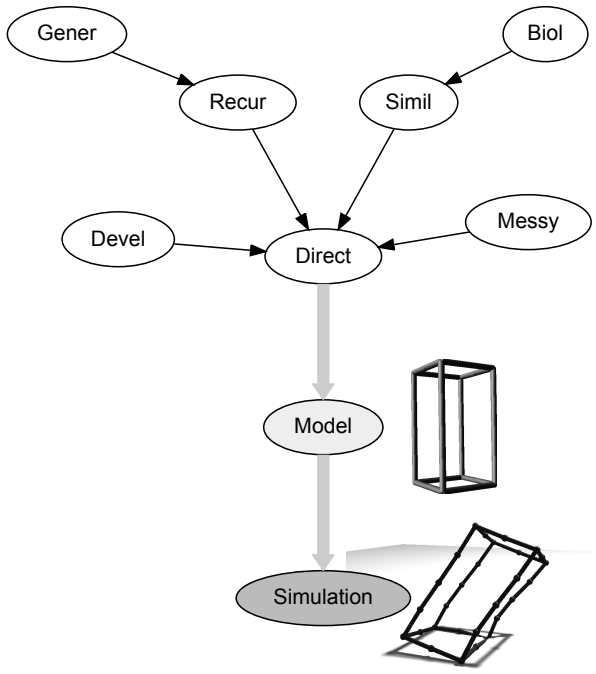

Figure 4: The idea of the hierarchy of genetic encodings.

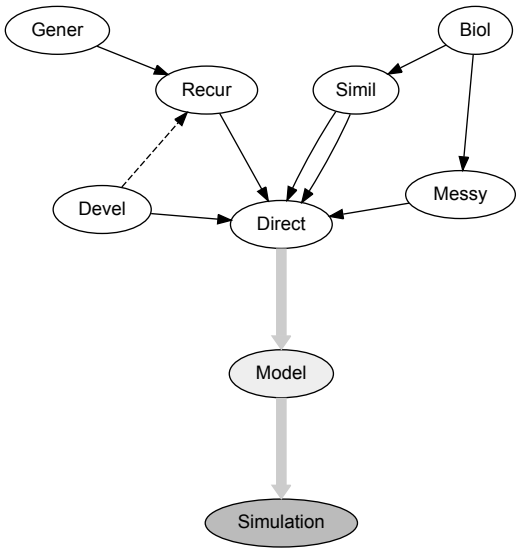

Figure 5: Multiple and alternative paths in the graph of genetic encodings.

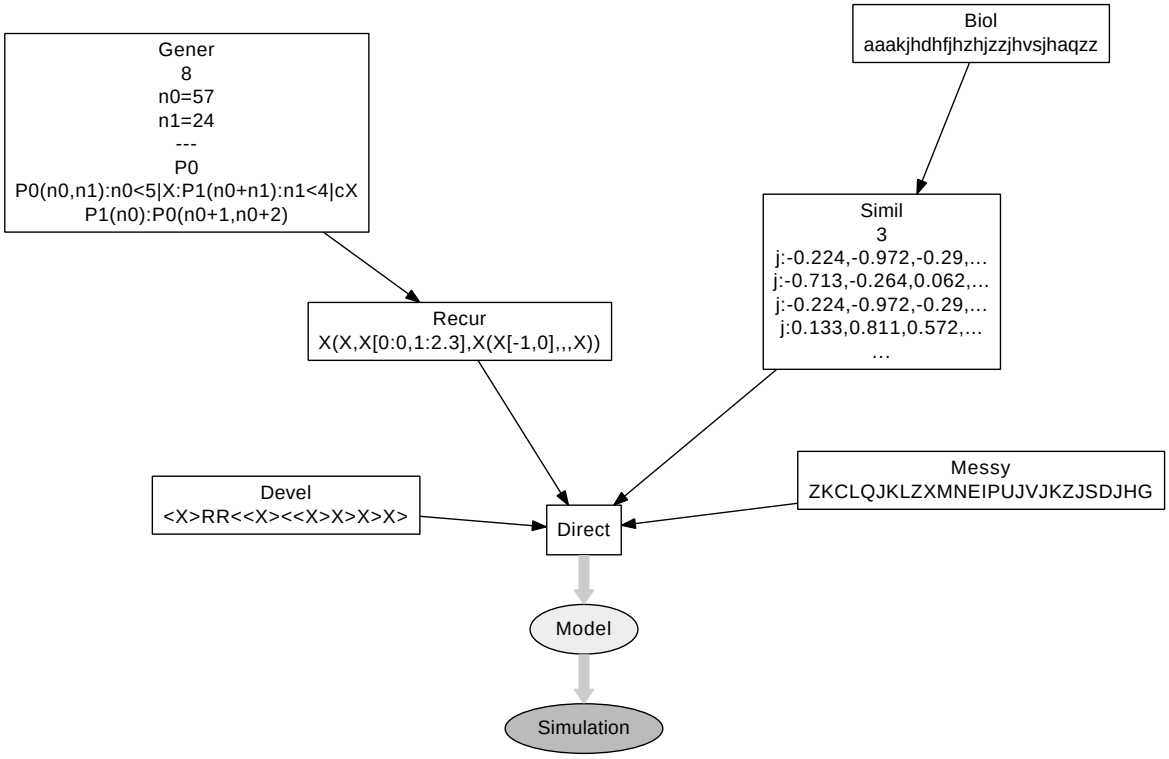

Figure 6: Genotypes require specialized genetic operators for each genetic encoding.

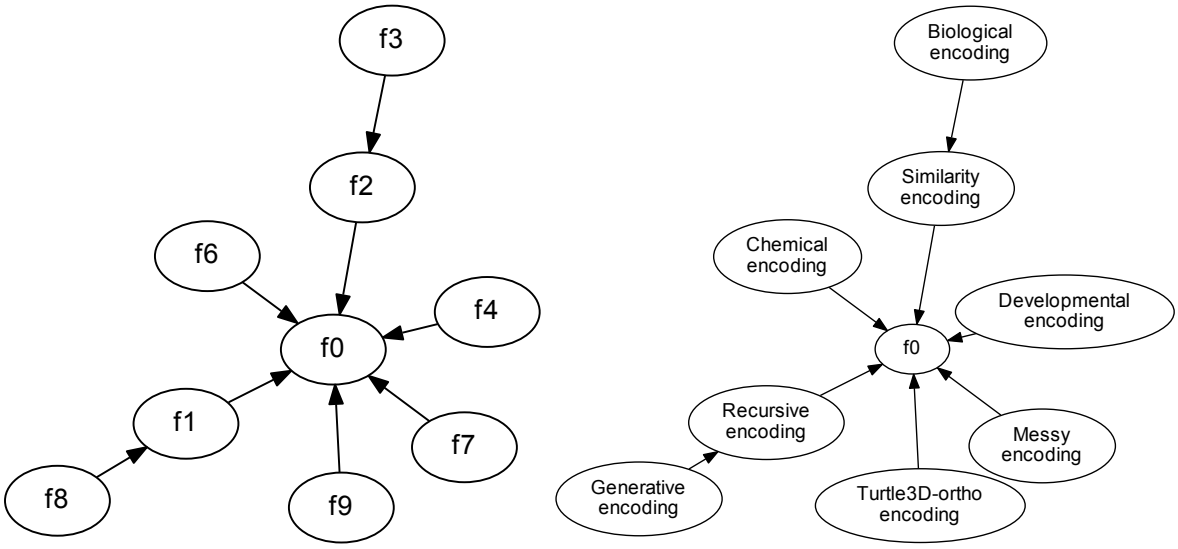

Figure 7: Genetic hierarchy with both symbolic and descriptive names of representations.

2.1 Description of genetic encodings

The following list outlines major properties and concepts of the eight encodings that have been used in experiments; sample genotypes are presented as well when they are one-line strings.

- *f0*: Direct encoding
  The genotype in this encoding enlists all elements of a phenotype. The particular format used here is not relevant, since mutation and crossover work on phenotypes, not on genotypes. Mutations change individual aspects of a phenotype (locations, connections, or properties of elements of the model), and crossover joins two halves of parent bodies that have been cut by a random plane [20].

- *f9*: Turtle3D-ortho encoding
  This extremely simple encoding uses six letters for up, down, left, right, forth, and back. The phenotype body is "drawn" in a 3D space using steps of unit length. Control system cannot be currently encoded. A sample genotype is `BBBRLLRUDDFUUDFFF`. Mutation changes, adds or removes one letter. Crossover is two-point.

- *f1*: Recursive encoding
  This encoding uses X characters to describe Joints, and neural units with their weighted connections are placed in square brackets. Parentheses mean branching of a structure. Additional letters modify length and other properties of body elements. A sample genotype is `XXrrX(X,LLLX[G:3])`. Mutations and crossover work on the textual representation of the genotype; crossover is two-point.

- *f4*: Developmental encoding
  This encoding may look similar to the *f1*, but it in fact describes a process of growth and specialization. There are three additional characters (genes) introduced. The '<' gene divides a "cell" into two cells, and the '>' gene finishes development of a cell. The '#' gene repeats a sequence of development encoded by the following genes. A sample genotype is `<L<XE#2>><WX>Xs>XI`. Mutations and crossover work on a tree structure that represents a program of growth (development) of a phenotype.

- *f2*: Similarity encoding
  In this encoding, each element of body and brain constitutes a separate entity that has two

"handles" represented by a vector of numbers. When the phenotype is constructed, these entities are joined so that the most similar handles stick together. Crossover picks random elements from parents and passes them on to offspring. Mutation adds elements, removes them, changes handles or changes properties of elements.

- *f3*: Biological encoding
  This representation encodes the concept of *f2* in the string of characters that constitutes a genome; each substring in the genome that starts with the "aa" codon and ends with the "zz" codon is assumed to define a single entity in *f2*. A sample genotype is `aabzzaabzzaabouhvpzz`. Mutation consists in substitution, deletion, insertion, gene duplication, or translocation. Crossover is either a "horizontal gene transfer" (a single gene is transferred from one genotype to another) or a standard crossing over where each gene from the parent genotypes is passed on to one of the children.

- *f7*: Messy encoding
  This encoding represents the idea of a "hash function"; the genotype is a string of capital letters that encode individual elements of body and brain. A small change in the genotype may result in unpredictable changes in the corresponding phenotype, so the mapping between genetic and phenetic spaces is disordered. A sample genotype is `BBAABAGABTZBVUOQ`. Mutations change random letters, and crossover is two-point.

- *f8*: Generative encoding
  This encoding allows to represent a parametric Lindenmayer system (L-system) [25, 10]; rules (productions) are used to generate a *f1* genome. One can think of this representation as a set of formal grammar rules that are modified and refined during optimization. During embryogeny (phenotype growth), the rules must meet some conditions to fire (the conditions are also a part of the genotype), and may call other rules or produce a part of a *f1* genome. Mutation methods are numerous; crossover produces one offspring that contains selected productions from both parents.

3 Experiments

Two kinds of experiments have been conducted to test performance of the eight encodings described earlier:

- Maximize vertical position of the center of mass.
- Maximize vertical position of the top vertex.

The task of maximizing height, used earlier by the author [20], is particularly suitable for testing performance in evolutionary design. It has only one, well-defined criterion, yet it requires complex structures to emerge that fulfill multiple requirements. This task is especially demanding where genetic encodings do not implicitly provide any domain knowledge on how the structures should be designed. The difficulty of building tall structures and opposing gravity is also manifested by theory, practice and expertise in engineering, design, and architecture.

3.1 Experimental setup

The experiments took approximately 3 weeks on a one-core 3 GHz CPU and were performed using the Framsticks environment [24, 23] with the following key parameters of the optimization process:

- optimization:
  - experiment definition: standard
  - gene pool capacity: 200

  - positive selection: steady state, tournament (2), mutated:crossed-over ratio 8:1
  - no multiple evaluation, no unchanged genotypes after selection
  - negative selection: two-stage – remove random, then remove worst ("boost phase")
  - stagnation: 3 000 or 10 000 non-improving evaluations
  - constraints: no more than 20 sticks in a structure
- simulation:
  - simulation period: 1000 time steps
  - initial elevation: 0.1
  - performance sampling period: 20
  - neural net simulation: off
  - performance calculation: after stabilization
  - sampling period while waiting: 100
  - allowed distance to be stable: 0.01
  - world: flat
  - simulation engine: MechaStick
  - gravity: 1
  - minimal/maximal joint length: 0/2
  - imperfection – initial movement: 0.01

The optimization process consisted of two stages. In the first phase, negative selection was random. Once the stagnation was detected, the negative selection was switched to "remove worst genotype in population", which resulted in a highly convergent optimization. When another stagnation was detected, the process ended. For the two fitness goals mentioned above, two kinds of experiments were performed with different stagnation periods. This results in 4 experiment setups, and for each setup, 10 independent optimization runs were performed for each of the 8 genetic encodings.

A fully deterministic and accurate simulation would result in "perfect" structures being discovered, i.e., ideally vertical bars that would stand vertically forever. To avoid this, some kind of randomness and noise must be introduced. This noise took the form of small, random forces applied to each vertex of a structure once the structure was created in the simulator. The structure was therefore shaken slightly, so that ideally vertical, unrealistic bars would fall down. However, adding random noise poses a problem: the simulation is no longer deterministic, and what was advantageous in eliminating unrealistic behaviors turns out to help other imperfect structures to gain fitness. Some structures could manage to stand straight *because* random forces were applied; these forces can accidentally help such structures gain high fitness.

To alleviate this problem, the evaluation of each structure should be performed multiple times, and its fitness should be averaged. Unfortunately, this does not fix the problem entirely. Due to the nature of the optimization process, even when the evaluation of each structure is repeated 2, 10, or 100 times, the selection mechanism will always prefer structures with better average fitness which means structures that were lucky in 2, 10, or 100 evaluations and the noise turned out to be favorable for them (more than for other structures). Increasing the number of evaluations increases the time needed for optimization drastically, yet it only reduces the problem of "fortunate winners, undeserving losers", not eliminates it – compare Fig. 12 in [20] and related discussion therein. Another issue is that non-determinism makes it impossible to directly answer the fundamental question of "what is the best solution found in optimization"; one can only say that the more evaluations, the more reliable the fitness. Also, in the steady-state architecture with cloning (i.e., multiple evaluation) enabled, the higher current fitness, the more evaluations – so the reliability of fitness varies for different genotypes.

Since the number of evaluations of different structures (the sampling of the search space) was the priority here, noise was introduced to eliminate perfect, unrealistic structures, but multiple evaluation

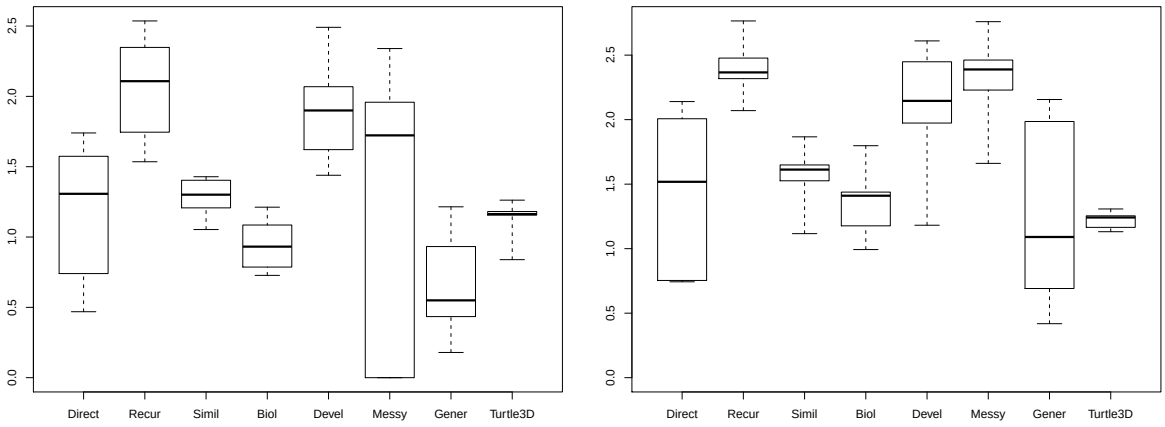

Figure 8: Best fitness in maximization of the vertical position of the center of mass. Stagnation period was 3,000 evaluations (left plot) and 10,000 evaluations (right plot).

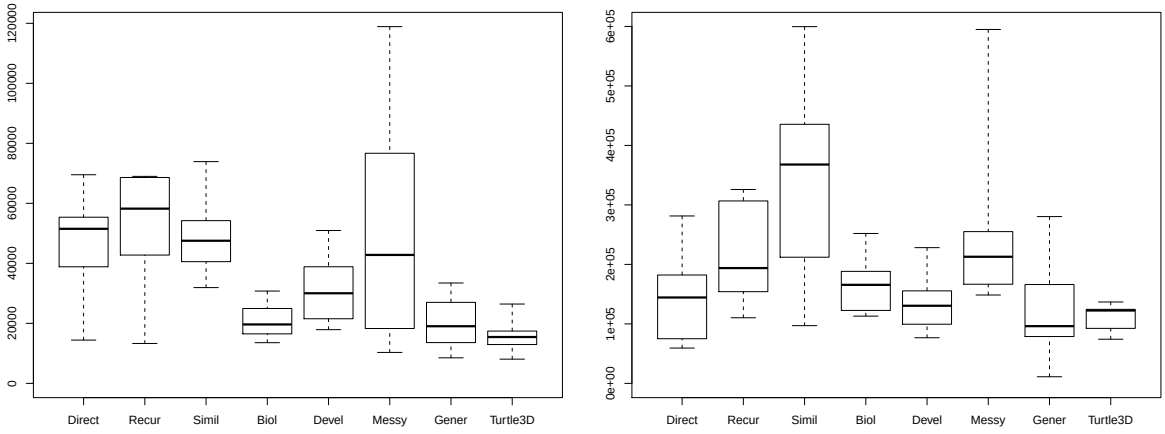

Figure 9: Total number of evaluations in maximization of the vertical position of the center of mass. Stagnation period was 3,000 evaluations (left plot) and 10,000 evaluations (right plot).

and averaging of fitness were not employed. This results in some fortunate structures being considered best, yet being unable to reproduce their behavior because it were random perturbations that helped these structures get high fitness when they were evaluated. You can see the manifestation of this phenomenon in the images shown in the Appendix: some of the best constructs turned over when they were simulated for the second time – they lacked (random) perturbations that allowed to keep them upright when they were evaluated during evolution.

3.2 Maximization of the vertical position of the center of mass

Figs. 8 and 9 compare fitness and number of evaluations in the maximization of the vertical position of the center of mass; each column in the box plot summarizes 10 runs of the experiment.

The fitness plots demonstrate that increasing the stagnation time from 3,000 to 10,000 evaluations did not change the general pattern of relations between representations. The results of the longer run are generally slightly better, but the difference is not dramatic. Longer runs resulted in a less variable fitness values within 10 repetitions of the experiment for each encoding. The numbers of evaluations compared between encodings do not vary much; however, the number of evaluations for

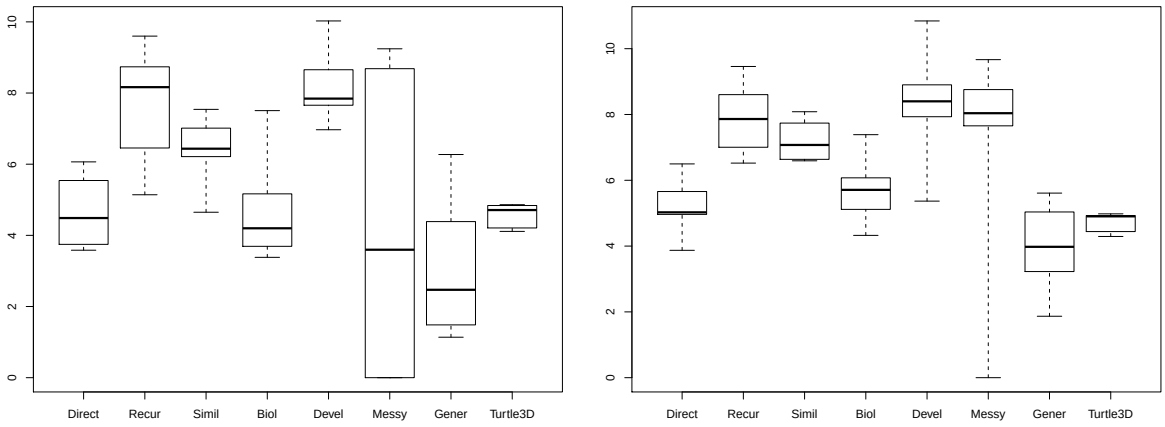

Figure 10: Best fitness in maximization of the vertical position of the top vertex. Stagnation period was 3,000 evaluations (left plot) and 10,000 evaluations (right plot).

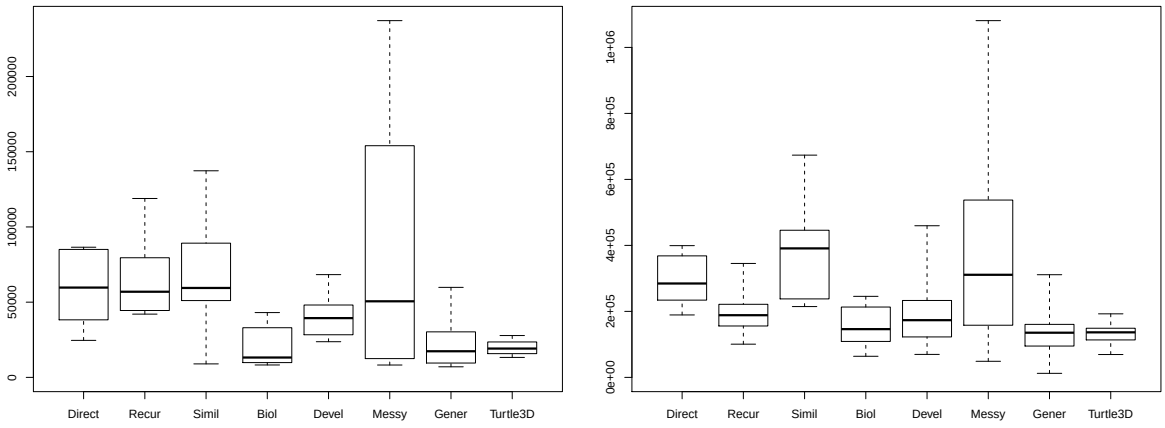

Figure 11: Total number of evaluations in maximization of the vertical position of the top vertex. Stagnation period was 3,000 evaluations (left plot) and 10,000 evaluations (right plot).

some encodings may be quite unpredictable (e.g., *Messy* or *Simil*). Elongating the period of non-improving evaluations from 3,000 to 10,000 (i.e., 3.33 times) resulted in a roughly similar increase in the total number of evaluations, even though a larger increase might have been expected.

3.3 Maximization of the vertical position of the top vertex

Figs. 10 and 11 compare fitness and number of evaluations in the maximization of the vertical position of the top vertex; each column in the box plot summarizes 10 runs of the experiment.

One can immediately notice that the fitness values are higher when vertical position of the top vertex is maximized; best constructs had their top vertex at the height of approximately 10, while best constructs in the previous task had their center of mass at about 2.5. Again, the results of the longer run are generally slightly better and the variability of fitness is smaller. As in the first optimization task, the number of evaluations is the least predictable for *Messy* and *Simil* encodings. Increasing the stagnation period 3.33 times yielded a similar increase in the total number of evaluations.

The actual designs that are outcomes of the optimization in both tasks are depicted in the Appendix.

4 Conclusions

Overall, in both optimization tasks, the general patterns and relations as demonstrated by performance plots are similar. Based on the results of the experiments it may be tempting to draw conclusions regarding comparison of efficiency of tested representations. These conclusions could be the following:

- The *Devel* representation provided nearly best results in a relatively short time.
- The *Recur* representation was almost as good.
- The *Turtle3D* representation yielded results that had the most similar (yet mediocre) fitness in both tasks.
- The *Messy* representation was the least predictable.

However, instead of comparing encodings in terms of better/worse, the results presented here should rather be treated as a demonstration of "this encoding can perform at least as good". The reasons are the following:

- Encodings *Direct*, *Recur*, *Gener*, and *Turtle3D* either allowed for restriction or were naturally limited so that the genetic operators would not produce structures with control systems. On the other hand, the four remaining encodings either have intrinsic abilities to produce neural networks and these abilities cannot be easily turned off, or were not restricted to only produce passive structures. Even though control systems were not simulated at all, this resulted in a much larger search space (a larger number of neutral mutations) for the latter encodings.
- Encodings *Direct* and *Turtle3D* allowed for cyclic (i.e., more rigid) structures; other encodings could only express tree-like structures.
- The *Turtle3D* encoding has a fixed, standard stick length (1), while other encodings can produce sticks of variable length (from 0 to 2). Given the limit of 20 sticks, fixed stick length directly influences the maximal height of constructs that can be designed.
- Some encodings – *Direct*, *Recur*, *Devel* – are considered more mature. More time has been spent on finding their weak points and improving them. The remaining encodings may carry some potential that is yet to be exploited.

Generally, in the two simple optimization tasks that have been considered in this report,

- All of the representations can be used depending on the requirements.
- No evidence was found that the complex generative representation *Gener* was advantageous.
- Reducing of the search space is beneficial.
- Numerous local optima exist.

Further work will concern improvement of the shortcomings that are already known for some of the encodings. Another direction of work is to provide more uniform conditions for the comparison – like the ability to remove the elements of the control system from the search space for all representations. Larger experiments (longer runs, bigger populations, more complex goals) and broader analyses of the data that describe dynamics of evolutionary processes will reveal characteristics, specifics and qualities of the encodings. For such analyses, measures estimating symmetry [13] and similarity [19] will be helpful; the latter will also allow for human-friendly visualization of results and investigation of properties of the search space for each representation [18].

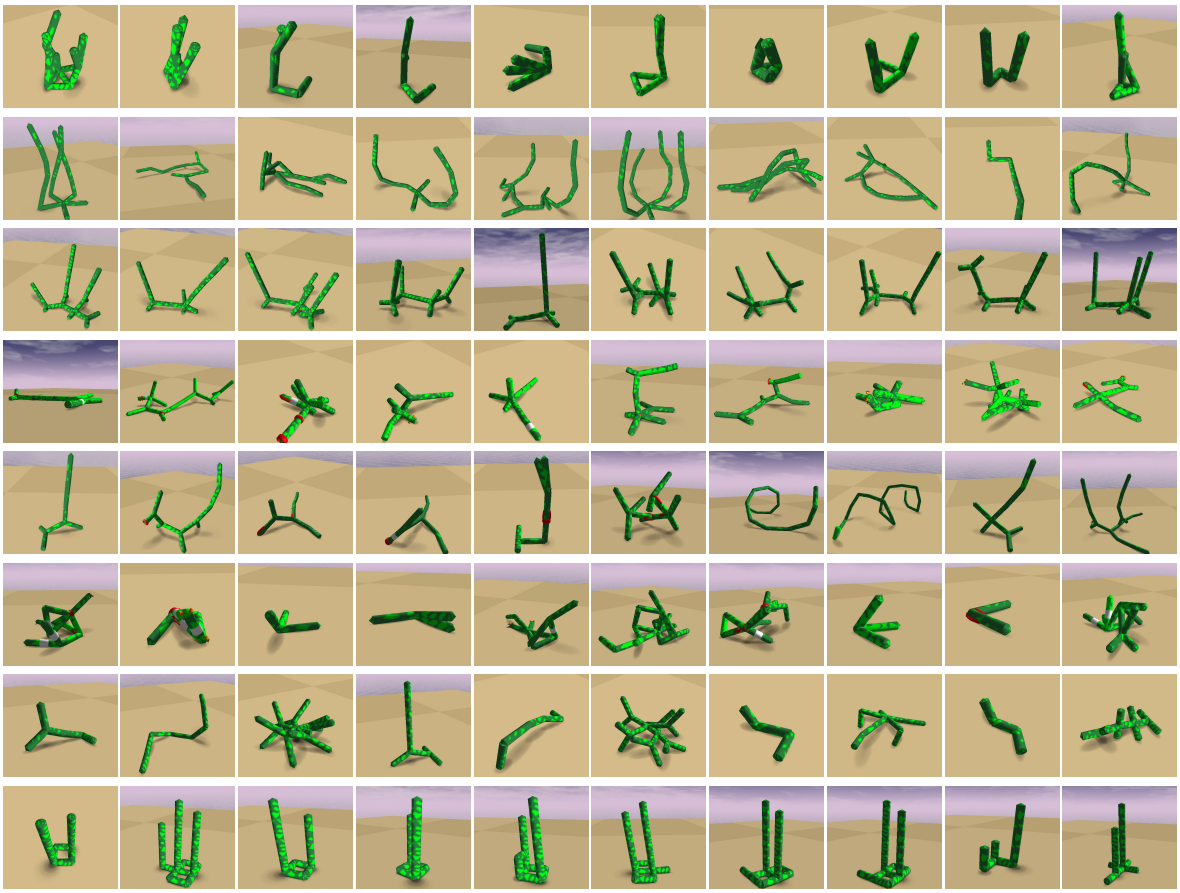

Figure 12: Maximization of the vertical position of the center of mass with the stagnation period of 3,000 evaluations. Encodings from top to bottom: *f0* (*Direct*), *f1* (*Recur*), *f2* (*Simil*), *f3* (*Biol*), *f4* (*Devel*), *f7* (*Messy*), *f8* (*Gener*), *f9* (*Turtle3D*).

Appendix

The following images show best constructs evolved for each genetic encoding in ten independent optimization runs. If viewing an electronic version of this report, zoom in to see more details. Figs. 12 and 13 concern maximization of the vertical position of the center of mass, and Figs. 14 and 15 concern maximization of the vertical position of the top vertex.

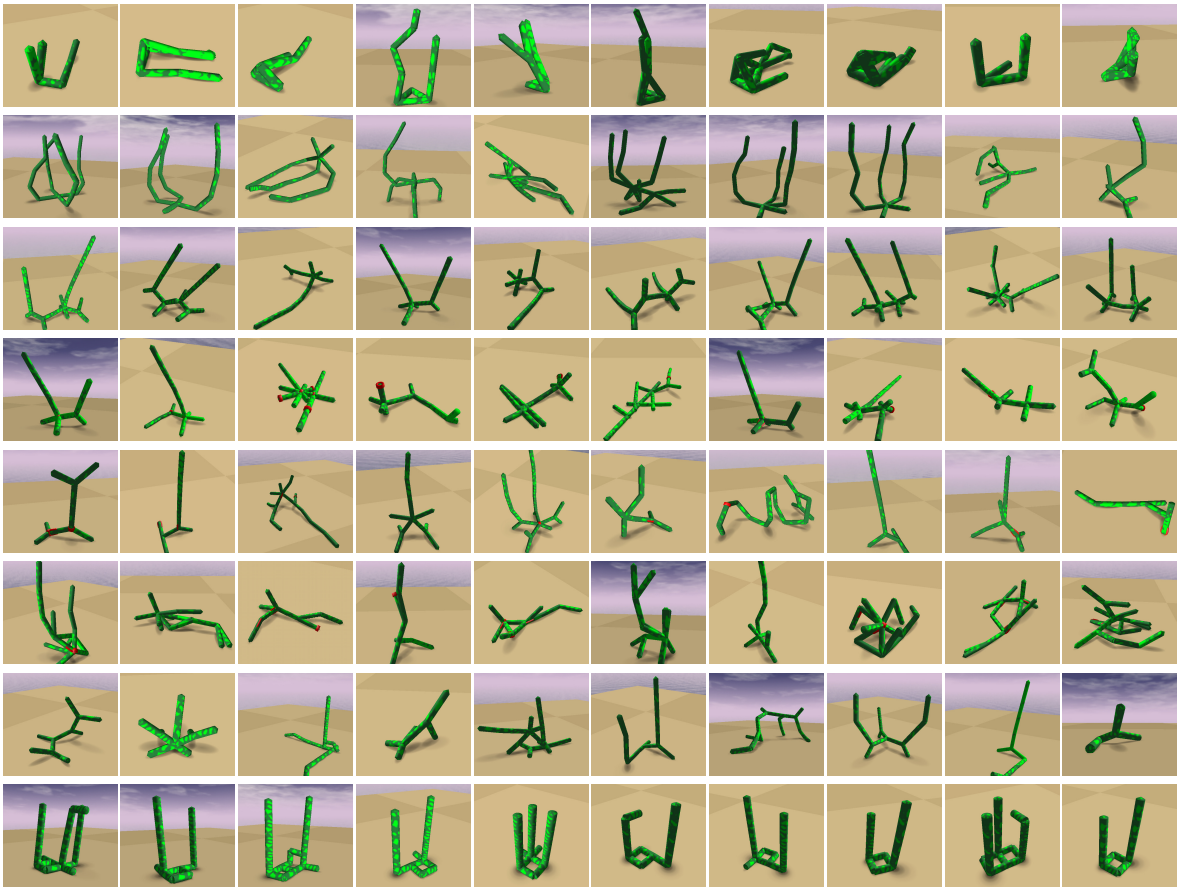

Figure 13: Maximization of the vertical position of the center of mass with the stagnation period of 10,000 evaluations. Encodings from top to bottom: *f0* (*Direct*), *f1* (*Recur*), *f2* (*Simil*), *f3* (*Biol*), *f4* (*Devel*), *f7* (*Messy*), *f8* (*Gener*), *f9* (*Turtle3D*).

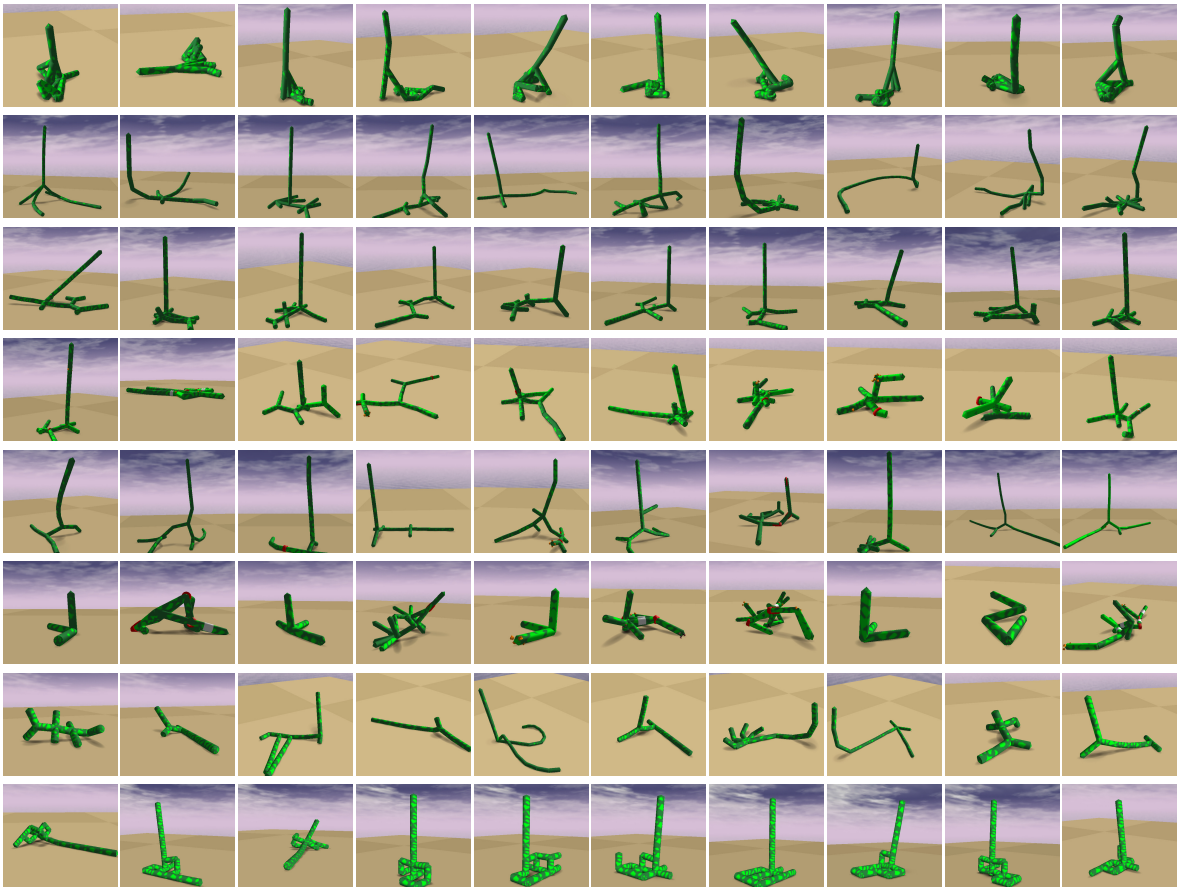

Figure 14: Maximization of the vertical position of the top vertex with the stagnation period of 3,000 evaluations. Encodings from top to bottom: *f0* (*Direct*), *f1* (*Recur*), *f2* (*Simil*), *f3* (*Biol*), *f4* (*Devel*), *f7* (*Messy*), *f8* (*Gener*), *f9* (*Turtle3D*).

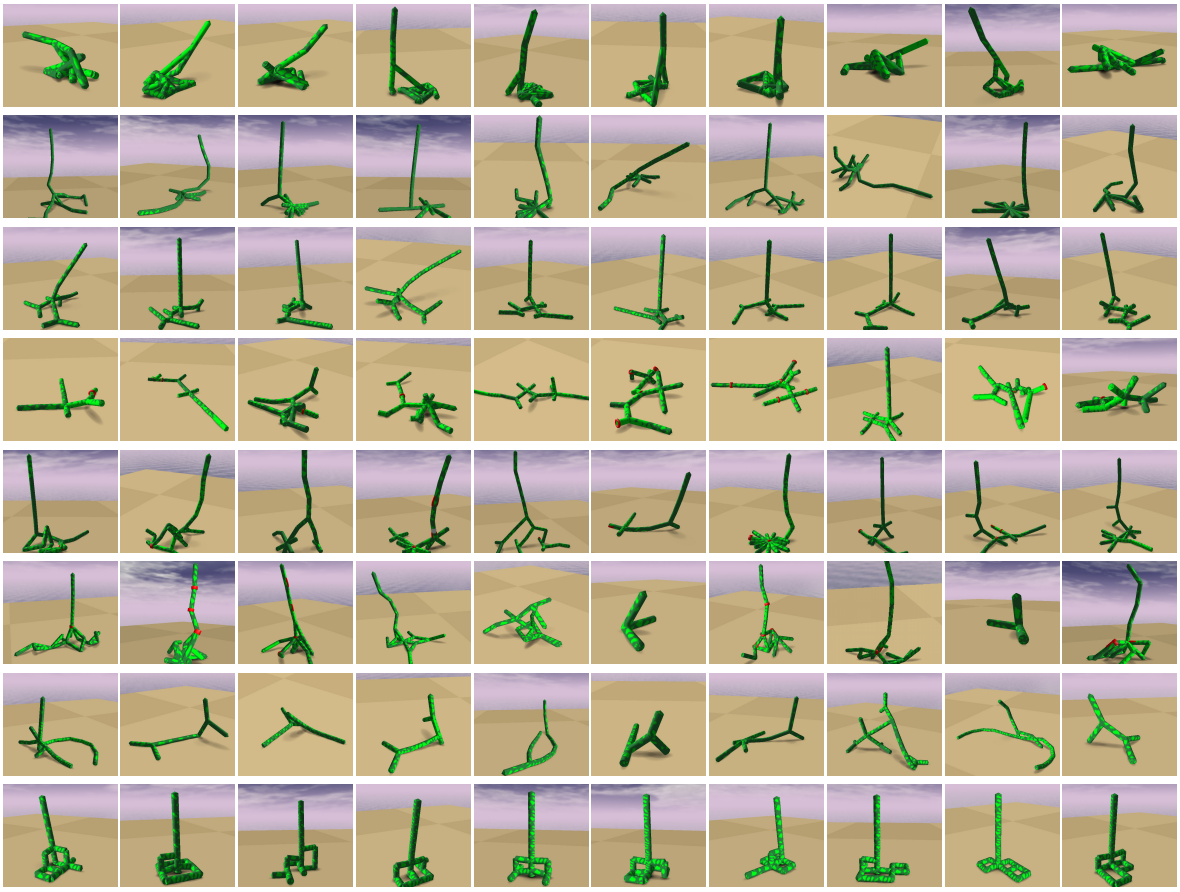

Figure 15: Maximization of the vertical position of the top vertex with the stagnation period of 10,000 evaluations. Encodings from top to bottom: *f0* (*Direct*), *f1* (*Recur*), *f2* (*Simil*), *f3* (*Biol*), *f4* (*Devel*), *f7* (*Messy*), *f8* (*Gener*), *f9* (*Turtle3D*).

References

[1] Peter Bentley. *Evolutionary Design by Computers*. Morgan Kaufmann, 1999.

[2] James M. Conrad and Jonathan W. Mills. The history and future of Stiquito: a hexapod insectoid robot. In Andrew Adamatzky and Maciej Komosinski, editors, *Artificial Life Models in Hardware*, chapter 1, pages 1–20. Springer, 2009. URL: `http://www.springer.com/978-1-84882-529-1`.

[3] G. F. Dargush and R. S. Sant. Evolutionary aseismic design and retrofit of structures with passive energy dissipation. *Earthquake engineering & structural dynamics*, 34(13):1601–1626, 2005.

[4] Frank Dellaert and Randall D. Beer. A developmental model for the evolution of complete autonomous agents. In P. Maes, M.J. Mataric, Jean-Arcady Meyer, Jordan B. Pollack, and Stewart W. Wilson, editors, *From Animals to Animats. Proc. of the 4th Int. Conf. on Simulation of Adaptive Behavior SAB '96*, pages 393–401, Cambridge, Mass., 1996. MIT Press.

[5] P. Eggenberger. Evolving morphologies of simulated 3D organisms based on differential gene expression. In *Proceedings of the Fourth European Conference on Artificial Life*, pages 205–213, 1997.

[6] James Foley. *Computer Graphics: Principles and Practice*, chapter 11. Addison-Wesley, Boston, 1997.

[7] Pablo Funes and Jordan B. Pollack. Evolutionary body building: adaptive physical designs for robots. *Artificial Life*, 4(4):337–357, Autumn 1998.

[8] Jacques Heyman. *The Science of Structural Engineering*. World Scientific Publishing Company, 1999.

[9] G. S. Hornby and J. B. Pollack. Creating high-level components with a generative representation for body-brain evolution. *Artificial Life*, 8(3):223–246, 2002.

[10] Gregory S. Hornby. Creating complex building blocks through generative representations. In Hod Lipson, Erik K. Antonsson, and John R. Koza, editors, *Computational Synthesis: From Basic Building Blocks to High Level Functionality: Papers from the 2003 AAAI Spring Symposium*, AAAI technical report SS-03-02, pages 98–105. AAAI Press, 2003.

[11] Gregory S. Hornby and Jordan B. Pollack. The advantages of generative grammatical encodings for physical design. In *Proceedings of the 2001 Congress on Evolutionary Computation*, pages 600–607. IEEE Press, 2001.

[12] A. J. Ijspeert, A. Crespi, and J. M. Cabelguen. Simulation and robotics studies of salamander locomotion. Applying neurobiological principles to the control of locomotion in robots. *Neuroinformatics*, 3(3):171–196, 2005.

[13] Wojciech Jaskowski and Maciej Komosinski. The numerical measure of symmetry for 3D stick creatures. *Artificial Life Journal*, 14(4):425–443, Fall 2008. URL: `http://dx.doi.org/10.1162/artl.2008.14.4.14402`, `doi:10.1162/artl.2008.14.4.14402`.

[14] V. G. Jáuregui. *Tensegrity structures and their application to architecture*. PUbliCan – Ediciones Universidad de Cantabria, 2010.

[15] R. Kicinger, T. Arciszewski, and K. De Jong. *Morphogenic evolutionary design: cellular automata representations in topological structural design*. Springer-Verlag, London, UK, 2004.

[16] R. Kicinger, T. Arciszewski, and K. De Jong. Evolutionary computation and structural design: A survey of the state-of-the-art. *Computers & Structures*, 83(23):1943–1978, 2005.

[17] R. Kicinger, T. Arciszewski, and K. De Jong. Evolutionary design of steel structures in tall buildings. *Journal of Computing in Civil Engineering*, 19:223, 2005.

[18] Maciej Komosinski. Applications of similarity measure in evolutionary optimization and design of 3D constructs. submitted.

[19] Maciej Komosinski and Marek Kubiak. Quantitative measure of structural and geometric similarity of 3D morphologies. *Complexity*, 16(6):40–52, 2011. URL: `http://dx.doi.org/10.1002/cplx.20367`, `doi:10.1002/cplx.20367`.

[20] Maciej Komosinski and Adam Rotaru-Varga. Comparison of different genotype encodings for simulated 3D agents. *Artificial Life Journal*, 7(4):395–418, Fall 2001.

[21] Maciej Komosinski and Szymon Ulatowski. Framsticks SDK (Software Development Kit). URL: `http://www.framsticks.com/sdk`.

[22] Maciej Komosinski and Szymon Ulatowski. Genetic mappings in artificial genomes. *Theory in Biosciences*, 123(2):125–137, September 2004. URL: `http://dx.doi.org/10.1016/j.thbio.2004.04.002`, `doi:10.1016/j.thbio.2004.04.002`.

[23] Maciej Komosinski and Szymon Ulatowski. Framsticks: Creating and understanding complexity of life. In Maciej Komosinski and Andrew Adamatzky, editors, *Artificial Life Models in Software*, chapter 5, pages 107–148. Springer, London, 2nd edition, 2009. URL: `http://www.springer.com/978-1-84882-284-9`.

[24] Maciej Komosinski and Szymon Ulatowski. Framsticks web site, 2019. `http://www.framsticks.com`.

[25] A. Lindenmayer. Mathematical models for cellular interactions in development i. filaments with one-sided inputs. *Journal of theoretical biology*, 18(3):280–299, 1968.

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

[27] J. D. Lohn, D. S. Linden, G. S. Hornby, and W. F. Kraus. Evolutionary design of an X-band antenna for NASA's space technology 5 mission. In *IEEE Antennas and Propagation Society International Symposium, 2004*, volume 3, pages 2313–2316, 2004.

[28] D. Marbach and A. J. Ijspeert. Co-evolution of configuration and control for homogenous modular robots. In *Proceedings of the Eighth Conference on Intelligent Autonomous Systems (IAS8)*, pages 712–719, 2004.

[29] Y. Rafiq, M. Beck, I. Packham, and S. Denhan. Evolutionary computation and visualisation as decision support tools for conceptual building design. *Innovation In Civil and Structural Engineering Computing*, pages 49–74, 2005.

[30] Stephane Sikora, David Steinberg, and Claude Lattaud. Integration of simulation tools in on-line virtual worlds. In Jean-Claude Heudin, editor, *Proceedings of 2nd International Conference on Virtual Worlds (VW2000), LNAI 1834*, Paris, France, July 2000, pages 32–43. Springer-Verlag, 2000.

[31] Karl Sims. Evolving 3D morphology and behavior by competition. In Rodney A. Brooks and Pattie Maes, editors, *Proceedings of the 4th International Conference on Artificial Life*, pages 28–39, Boston, MA, 1994. MIT Press.

[32] T. Taura and I. Nagasaka. Adaptive-growth-type 3D representation for configuration design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 13(3):171–184, 1999.

[33] V. Toğan and A.T. Daloğlu. Optimization of 3D trusses with adaptive approach in genetic algorithms. *Engineering structures*, 28(7):1019–1027, 2006.

[34] Kai Xu, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Fit and diverse: set evolution for inspiring 3D shape galleries. *ACM Trans. Graph.*, 31(4):1–57, July 2012. URL: `http://doi.acm.org/10.1145/2185520.2185553`, `doi:10.1145/2185520.2185553`.