Artificial Ontogeny for Truss Structure Design

Alexandre Devert, Nicolas Bredeche, Marc Schoenauer

Artificial Ontogeny for Truss Structure Design

Alexandre Devert, Nicolas Bredeche, Marc Schoenauer

***Abstract*—This paper introduces an approach based on Artificial Embryogeny for truss design to address the problem of finding the best truss structure for a given loading. In this setup, the basic idea is to optimize the size and length of beams in a truss through the actions of a set of cells that are distributed over the very truss structure. Given information at the mechanical level (beam strain), each cell controller is able to modify the local truss structure (beam size and length) during a developmental process. The advantage of such a method relies on the idea that a template cell controller is duplicated over all cells, keeping the optimization search space very low, while each cell may act in a different manner depending on local information. This approach is demonstrated on a classical benchmark, the "cantilever": resulting organisms are shown to provide very interesting and unique properties regarding reuse of optimized genotypes in noisy or higher-dimension settings.**

***Index Terms*—Artificial Ontogeny, Multi-cellular Model, Artificial Evolution, Truss Structure Design, Spatial Organization.**

I. Introduction

Indirect representations in Evolutionary Computation have been advocated for a long time, to cope with large highly constrained phenotypic spaces. Using a complex *ontogeny* (transformation that maps a genotype onto a phenotype) allows the exploration and optimization to take place in a lower dimensional space, and the constraints to be explicitely handled in the ontogeny process itself. Artificial Embryogeny (AE) is concerned with ontogenies involving an "embryo" (either user-defined, or evolved within the genotype) that undergoes a series of transformations that are encoded in the genotype, the *development stage*, ultimately leading to the phenotype. In most case, the development stages imply distributed computation between spatially distributed cells that interact with one another and (possibly) with the environment. This phenotype is evaluated once the development is ended and gives its fitness to the genotype. Evolution thus optimizes the development process rather than a solution of the problem at hand.

However, there is not yet any consensus about how to encode a development process. While rewriting rules were among the first proposed encodings [9], another popular encoding is *Cellular AE* (CAE), that borrows to the Cellular Automata paradigm: phenotypes are cells that locally interact. All cells share the same update rules, but have their own internal state. The cells, in an initial prescribed state, form the embryo, and the optimization is concerned with the common update rules. Early works on this idea include P. Eggenberger 3d shapes [6], where the cells are embedded in a grid and are controlled by a model of Gene Regulatory Network. J. Bongard [3] uses similar ideas to evolve virtual creatures. More recently, new ideas for CAE developped after the introduction of the "flag" testbed by J. Miller [15]: each cell is an element of a 2d square grid, and the goal is that each cell takes a color matching that of a target picture. In those approaches, update rules (or controllers) have been encoded in various ways, from boolean logic functions [15] to sets of rules [2] and various flavours of neural networks [8], [4], [5]. Taking a step further, Estévez et al. [7] demonstrate a CAE model that is able to produce designs of self-shadowing shapes thanks to feedback from the environment. While their approach focuses on evolving simple growing rules in a discrete world comparable to a 2D Cellular Automata, it demontrates the ability of developmental systems to address a class of problems rather than a single specific problem.

This paper aims at demonstrating that those ideas can be applied to a practical mechanical design problem, that of truss structures. Section II sets up the background, recalling the basic issues in truss structure design, and surveys previous evolutionary approaches in that domain as well as describing a preliminary experiment based on a direct approach to solve a specific instance of a typical 2D cantilever benchmark problem: the "cantilever" design problem. Then, an original CAE model for truss structures is proposed in Section III: beams and joints are encoded as two types of cells, on a predefined regular grid, and Finite Element Analysis embedded in the development process provide the beams with environmental feedback. Section IV presents the first results on the "cantilever" design problem, demonstrating the flexibility and robustness of the proposed approach, implying that our approach is able to solve a *class* of design problems rather than just *a single* specific instance. As usual, Section V concludes the paper by sketching on-going and further directions of research.

II. Truss Structure Design

A. The direct problem

Truss Structures are ubiquitously used in architecture and engineering, both in real-world applications and for prototyping purposes. Although being simple to represent and evaluate, they can be very complex, and are still a challenge for Optimum Design. The main goal when designing a truss structure is to make it as light as possible while being strong enough to resist a given situation, i.e. having as small deformations and stress as possible when some given loads are applied. Solving the inverse problem (what is the best structure for a given loading?) relies on being able to solve the direct problem (what are the deformations and stresses of a structure for a known loading?).

Trusses are made of beams connected by joints. There are several possible models for a beam, both in 2 and 3

TAO project-team, INRIA Saclay – Île-de-France and LRI/CNRS, Bât 490, Université Paris-Sud, 91405 Orsay Cedex, France

dimensions, depending on the mechanical phenomena that are taken into account in the modelisation and on the physical properties of the material the beams are made of. In this work, the simplest 2d bar model with linear material will be considered: the effect of flexion are ignored, a beam can transmit efforts only by compression or elongation, and the stress is proportional to the strain. Moreover, the weight of the structure is neglected. A beam is hence modeled as a rigid spring, whose stiffness is the product of its cross-section area and the Young modulus of its material. In this planar model, each joint has at most 2 degrees of freedom.

From there on, solving the direct problem using a Finite Element Method (FEM) is straightforward: the unknown variables are the displacements of all joints, the sum of all forces applied to a given joint is 0 at equilibrium, and each beam results in a linear equation involving the displacements of its 2 joints and the forces applied there.

The matrix associated to each beam is called *local stiffness matrix*. Those matrices are then *assembled* to form the *stiffness matrix* K, summarizing the whole truss equilibrium. K is positive-definite, symmetric and usually very sparse, and hence solving $KU = F$ is straightforward (F contains the load for each joint), even for large sizes of K. From the joints displacements U, one can deduce the beams strains and stresses.

B. Optimization of truss structures

While the direct problem can be considered solved by FEMs, there is not yet a general consensus on how to solve the inverse problem, and optimize a truss structure for a given situation. There are three kinds of truss design problems. The *sizing problem* tunes the cross-sections of beams on a fixed truss geometry. The *configuration optimization* additionaly moves the positions of the joints (and hence the beam length) for a fixed topology (i.e. the number of trusses and their connections are fixed). Both problems deal with a fixed number of design variables, generaly continuous (though the cross-sections can be restricted to practically available discrete values). The *topology optimization* consists in finding an optimal connectivity of the beams, and how to represent a general truss structure remains an open issue.

With their ability to cope with any representations without making strong assumptions about the fitness function, evolutionary algorithms has been popular for truss structure design since their early years, as recently surveyed by R. Kicinger et al. [12]. Most of these early works focus on a direct representation of the truss structure, where the genotype is a list of design variables. However, more recent works are amenable to Artificial Embryogeny: R. Kicinger and T. Arciszewski [11], [13] propose to evolve the rules of a 1D discrete cellular automaton to gradually build, floor by floor, the structure of a whole building. Going one step further, T. Kowaliw et al. [14] optimize a set of rules for a 2D discrete cellular automaton that builds the truss structure. This mapping allows to evolve bigger structures than any of the previous works, and display an interesting feature as the best genotypes can be reused in different contexts, without having to re-run the optimization anew.

C. Solving the Inverse Problem with a Direct Approach

In order to illustrate the problem of finding the best sizing and configuration for a given truss structure, this section describes a direct approach to solve a specific instance of this problem. In this setup, all beam radii and joints 2D coordinates are optimized thanks to an evolutionary algorithm. The problem considered is that of a cantilever, for which the geometrical and mechanical conditions for two different examples are shown on Figure 1: two joints are fixed on the right, and some force is applied half-way of the vertical left boundary. The goal is to minimize the weight of the truss while maintaining beam stress (in fact, strain here, as both are linearly correlated) within a given range. Practicaly, each beam is supposed made of steel (density 7860 kg/m^{-3}; Young modulus $210 \times 10^9\ N/m^2$). In this preliminary experiment, both the $4*4$ and $8*6$ cantilever problems are considered and the initial guesses for both truss structure (ie. genome boostrap values) are given in figure 1.

The phenotype, ie. the truss configuration, is directly build from the genotype, which contains the list of each node coordinates and beam sizes. From the resulting truss, a single FEA is performed to ensure that none of the beam strains is over a maximum value (0.1), otherwise a fitness equals to the weight of the embryo is returned. If not, the weight of the actual truss structure is considered as the fitness.

The optimization process is performed using the state-of-the-art CMA-ES algorithm [1] with its default parameters. Experiments are run for (at least) 50000 evaluations, and results are statistics over 96 independent runs. Genotype sizes are 100 (resp. 278) for the $4 * 4$ (resp. $(8 * 6)$) geometries depending on the number of parameters. Figure 2 shows the evolution curve, which converges toward individuals with a stable fitness in 50000 evaluations, and the phenotype built from the best individual. Interestingly enough, most of the best individuals feature few, if not none, modification of the original beam lengths and focus only on beam radii as is shown for the best phenotype for the 8×6 problem (darker beams have a bigger diameter).

While this direct approach provides optimized individual for a given setup, it fails to be re-usable as is and the whole optimization process must be restarted whenever a parameter of the problem is modified. Moreover, genome size strongly depends on the truss size, which is a key problem for generating large size structure. In order to address these issues, an approach based on artificial ontology is decribed in the next sections.

III. Truss Ontogeny: The *EiffelBlob* Model

EiffelBlob, the original approach proposed in this paper, is based on a CAE model for a truss structure, and an evolutionary model for the optimisation of its update rule. More precisely, genotypes are the weights of a Multi-Layer Perceptron with fixed topology, and phenotypes are truss structures, whose fitness is computed based on their mechanical behavior. The evolutionary part is hence straightforward, whereas the originality of *EiffelBlob* lies in the CAE model, involving in particular mechanical feed-back from the mechanical behavior of the structure on its geometrical layout.

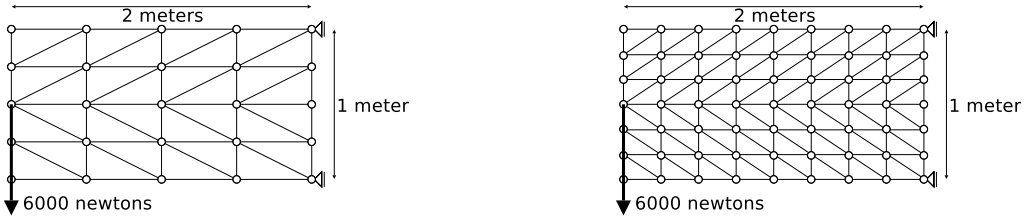


Fig. 1. The initial trusses used to bootstrap the development process, for respectively *4×4* and *8×6* cantilever problem.

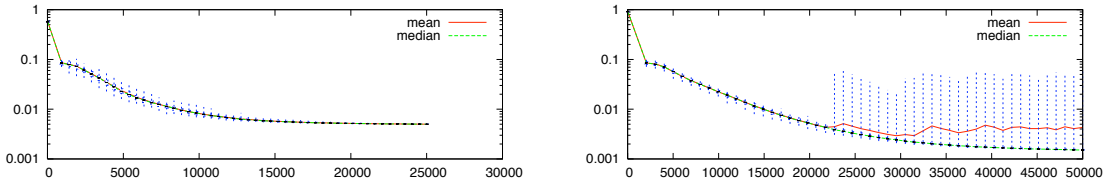


Fig. 2. Results for the *4×4* (left) and *8×6* (right) cantilever problems ; X axis is number of fitness evaluations, Y axis is weight with log scale.

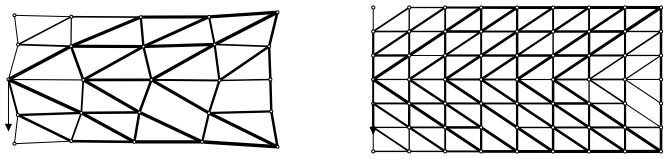


Fig. 3. Best phenotype for the *4×4* (left) and *8×6* (right) cantilevers with the direct approach. Darker beam means bigger cross-sectional area (ie. bigger diameter).

A. The development model

Both beams and joints are considered in *EiffelBlob* as *cells*, i.e. entities having both an internal state (a vector of real values, termed *chemicals*) and an update rule, or controller. The controller takes inputs from the internal states of the cell and all its neighbors, and from the outside world. Its outputs are some updates of its internal states, and some commands that will trigger physical alterations of the truss element it controls. New external inputs are then gathered, and the process iterates until some stopping criterion is reached. The final state of the truss is the phenotype corresponding to the genotype, and its mechanical properties determine the fitness of this genotype.

The complete truss is called the *organism*, and can be seen as a graph with two kinds of cells: the joint cells and the beam cells. Because this study is restricted to planar trusses made of triangle sub-elements, each joint is connected to at most 6 beams, and each beam is connected to exactly 2 joints. Figure 4 gives an example of the neighborhood relationships between different cells of a simple truss.

Beam cells act as both sensors and effectors of the truss organisms since they get some feedback from the environment through accessing information about physical beam strain (see Section III-D) and trigger local alterations of the truss: increase or decrease of both the length and the cross-section of the beam (see Section III-C). All beam (resp. joint) cells share the same controller, while having their own internal state. Hence, the development of the organism is completely determined by the initial truss and one controller for each type of cell. In particular, the complexity of the genotype does not depend on the complexity of the initial truss.

B. The controllers

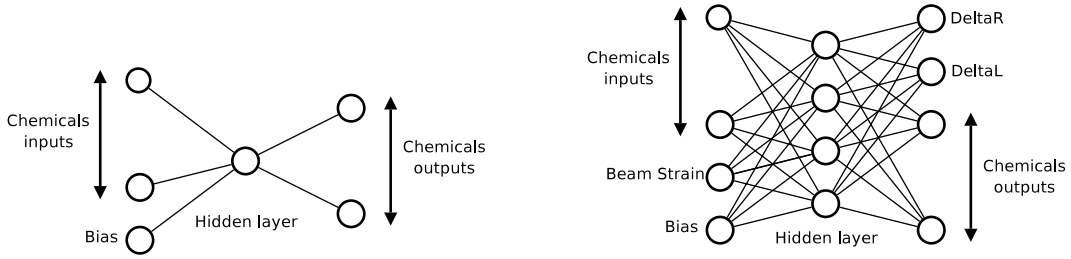
Figure 5 shows both the joint and beam cell controllers architecture. A joint cell can be seen as transmitting information among all its neighbor beams. Beam cells have additional inputs and outputs, respectively a mechanical feed-back from the Finite Element Analysis (see Section II-A), and the two outputs termed Δ_L and Δ_R, that provide the controller the ability to modify the truss geometry (see next Section).

Practically, controllers considered here are simple perceptrons with a fixed architecture[1] (a single hidden layer made of 4 neurons for the beams, a single hidden neuron for the joints, as shown on fig. 5). Neurons from the hidden layer and the cell state update output use the classical $tanh$ activation function, whereas the Δ_L and Δ_R outputs of the beam controller use a modified activation function (namely, $\sigma(x) = 0.282842\ x\ e^{-x^2}$) to ensure a smooth modification, even with completely random controllers.

The key issue in our approach is that controllers over the truss structure are homogeneous: all joint (resp. beam) controllers share the same neural network parameters all over the truss structure, while each controller differs in its functional behavior depending on the context. As a result, it makes it possible to optimize a rather small number of parameters (the weights for the template joint and beam controllers) which is

[1]Preliminary experiments showed that this architecture is more than enough to study dynamics of the truss as recurrence may be implicit thanks to the truss topology.

Fig. 4. On the left, a simple truss. On the right, the corresponding cells: beam and joint cells are represented respectively by big white discs and little gray discs. The edges shows the neighbourhood relationships.



a – A joint controller

b – A beam controller, with one additional input and two additional outputs

Fig. 5. Sample joint- and beam-controllers as multi-layered perceptrons. The number of hidden units shown here may vary in practical and was chosen from preliminary experiments (not shown here). Note that any other kind of neural networks (e.g. Echo State Network) could be used as long as the number of inputs and outputs are respected. Yet again, it is important to note that recurrence is inherently possible while relying on simple multi-layered perceptrons due do the very topology of the truss structure, which can be seen in its whole as a recurrent neural network.

independent from the actual instanciated number of controllers over the truss structure.

C. Geometry control

At each iteration of the development, the controller for each beam provides values for Δ_L and Δ_R, the geometrical commands for the underlying beam. The length and the radius of the beam are respectively updated by adding Δ_L and Δ_R to the current length and radius of the beam. However, some geometry reparing may be necessary to avoid infeasible truss, for example when concurrent updates of beam lengths result in a violation of the triangular inequality. This is done globally, by considering the truss as a mass-spring system where each joint has unit mass and each beam is a spring with unit stiffness and rest length equal to the virtual length provided after variation due to the controllers. This mass-spring system is then relaxed until it reaches an equilibrium state, which is performed using a damped Verlet integration scheme, as in [10], with an integration step $\Delta t = 10^{-2}$ and a damping factor $\alpha = 10^{-2}$. The computational cost of this integration scheme is close to that of the explicit Euler scheme, albeit with a greater stability and a practical handling scheme of geometrical constraints management (e.g. fixed positions of some of the joints, sliding conditions, contact conditions on other structure, etc), especially when it comes to take into account constraints of engineering problems to be considered. Note that this procedure produces an implicit but global feedback to the cell controllers from the environment geometry, without any additional input.

D. Mechanical feedback

Right after the geometry update, a structural FEM analysis is processed as described in Section II-A. The corresponding linear system is solved by a standard iterative solver (here: a Conjugate Gradient algorithm with Jacobi preconditioning). Iterations are stopped whenever the residual norm is below 10^{-6}. Even though the algorithm theoreticaly converges in at most N steps (with N is the size of the matrix), it might in practice take much longer, depending on the condition number of the linear system. Hence an absolute limit on the number of iterations is also set $(20 \times N)$, and the truss is considered invalid if the residual norm is still too high after that many iterations. In this latter case, development stops and a strongly penalized fitness value is returned for this truss. On the other hand, in the standard case, strain values obtained after convergence are used as inputs of the beam controllers and development goes on.

E. Developmental Process and Fitness Computation

The initial condition of the developmental process considers that all chemicals for communication between cells are set to zero and that the truss is set to a default initial position. This position, while clearly sub-optimal with regards to the problem at hand, represents an opportunity for bootstrapping the optimization process. Considering this, the initial configuration setup used in the following experimental sections is considered as shown on figure 1, which is a relevant, yet far from optimal, starting point for the cantilever problem at hand.

The problem of halting the development process must be considered very carefully. Indeed, in previous work [4], the authors demonstrated, on a CAE approach to the classical flag problem, that the choice of the stopping criterion is a key issue regarding solution robustness in CAEs, and that it is more efficient to wait for the stabilization of the process onto a fixed point than to stop it abruptly after a pre-defined fixed number of iterations. Such a stopping criterion is implemented here by monitoring a measure of the variance, over a sliding time

window, of the global energy of the organism, computed as the L_2 norm of the array of cell state vectors. The development stops whenever the variance of this energy is smaller than 10^{-15} in any window of 16 consecutive time steps. A penalty is added to the fitness of the individuals whose development didn't stop according to the energy criterion within a user-defined total number of iterations (2048 in all experiments here), and the organisms whose development didn't stop afer twice this number of iterations get very poor fitness without any Finite Element Analysis.

If both the geometrical process (Section III-C) and the development process above end up successfully, the fitness is computed as the weight of the truss (hence depending on the radii and lengths of all beams) penalized by a quadratic penalty depending on violation of the constraint on the strain.

IV. Experimental results

The *EiffelBlob* model has been evaluated on the same classical *cantilever* benchmark problem described in section II-C. However, it differs from this preliminary experiment as the development of the *EiffelBlob* lasts for several iterations (see Section III-E). Once development is over, fitness is computed in a similar fashion, by only considering this final stable configuration.

While the fitness function is the same for both the direct and *EiffelBlob* approaches, the number of unknown variables in *EiffelBlob* is constant wrt. the truss size for the CAE approach as it is concerned only with with the controller parameters (genome size of 41 in this case, encoding the weights of the template controllers which are duplicated over the truss structure, see section III-B). As in the preliminary experiment, optimization is performed using the state-of-the-art CMA-ES algorithm [1] with its default parameters, except for the initial mutation step-size which is set to 10^{-2}. All experiments are run for (at least) 10000 evaluations, and results are statistics over 20 runs.

Figure 6 shows the results for the two benchmark problems: the 4×4 and 8×6 cantilevers. A first remark is that the indirect method converge towards a comparable fitness value with that of the direct approach (cf. section III-E) for both problems. However, the indirect encoding approach converges with much less iterations of the optimization algorithm, and using approximately the same number of iterations for both problems (slightly less than 10000). On the opposite, the direct approach requires a number of iterations that increases with the number of variables, as expected. It should be noted, however, that those plots don't show the actual computation time: one single evaluation of the indirect approach involves up to ≤ 4096 FEAs, to be compared with the single FEA that is needed by the direct approach. Yet again, the direct approach does not provide solution that can be re-used in a different setup, which is the key motivation for this work - while the *EiffelBlob* model do actually feature such a capability, as well as being able to recover from perturbated initial condition. This is shown in next section.

A. Truss design and reusability

Indeed, the key feature of the indirect approach relies in its ability to provide robust and reusable controllers. Three different scenarios have been set up to validate this claim. In the *Perturbed Geometry* scenario, 10 different initial geometries are used, where the joint positions in the 8×6 truss (Figure 7) are randomly drawn within a disc of radius 0.05 around their initial position. In the *Load Increase* scenario, the 8×6 cantilever is now considered with a load of $12000N$ instead of $6000N$. And in the *Larger Dimensions* scenario, a 9×6 (regular) initial geometry is used instead of the initial 8×6 cantilever.

The best individuals for each of the 20 runs of the 8×6 cantilever problem presented in previous section are re-evaluated using these perturbed experimental conditions, *but no further evolution process occurs*. The development is performed during at most ≤ 4096 steps. The resulting fitness is then compared to that obtained by optimizing with the direct approach: for each scenario, the direct approach is run 20 times from the same geometries for 50000 iterations so as to serve as a performance reference to evaluate adaptation capability of the *EiffelBlob* individual.

Table 4.2 shows the results for each of the problems, with the following comparison protocol: given a developped *EiffelBlob* individual, the weight obtained after development is compared to that obtained after optimization by the direct method for the same problem. If the beam stress is above the threshold, the CAE solution is considered invalid; if it is valid, if the weight obtained by the *EiffelBlob* method is lower or equal than that of the direct method (up to $2\times$ the standard deviation of the weight obtained by the 20 direct approach run), the CAE solution is considered 'optimal' (and suboptimal otherwise).

While results do depend on the actual perturbed scenario (deeper studies revealed that some invalid results were due to geometry implementation, which did not handle well very flat triangles and resulted in instable mass-spring system), a vast majority of individuals did achieve valid, if not optimal, results. Moreover, individuals with perfect development are rather frequent, and advocate the relevance of the developmental approach with regards to robustness towards noisy initial conditions and truss size scalability: to some extent, less than 4096 FEAs are needed to develop a truss in a novel setup (and very often much less), while close to 50000 FEAs are needed for the direct approach whenever a new experimental setup is presented (see section II-C).

V. Discussion and Conclusions

This paper introduced *EiffelBlob*, an original developmental model for truss design. The problem addressed is to minimize the total weight of a given truss structure, given that evaluation of the total weight is computed only once (ie. a black-box optimization scenario). *EiffelBlob* performs actions at a geometrical level, where a simple spring simulation is used to expand the structure in a 2D space, relying on feedback information from the environment obtained at the mechanical level (using FEM). In practical, cells are distributed over the

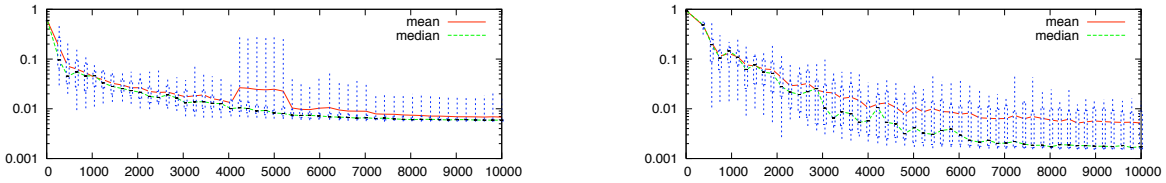


Fig. 6. *EiffelBlob* indirect approach. Left: *4×4* cantilever ; Right: *8×6* cantilever. X axis is number of fitness evaluations, Y axis is weight with log scale.

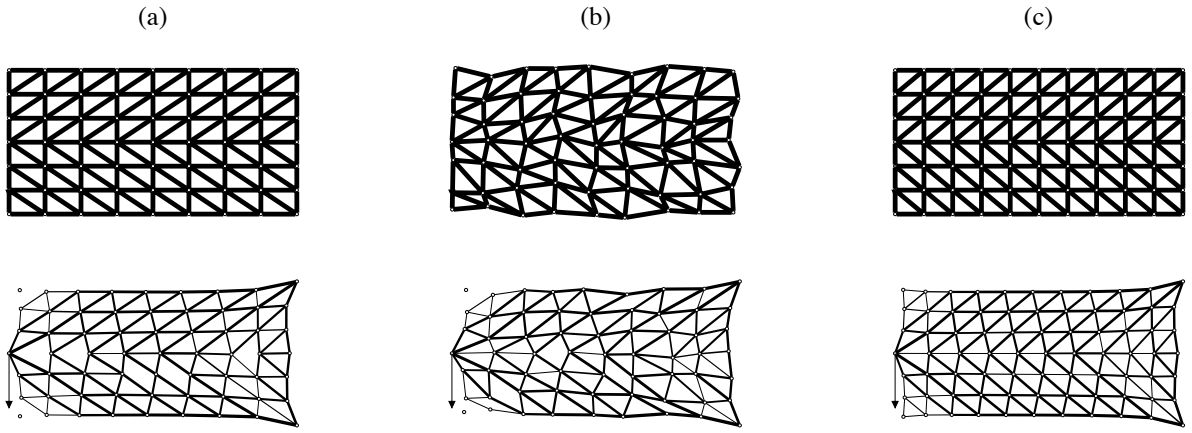


Fig. 7. initial (1st line) and final (2nd line) configuration for (a) standard 8×6 truss (b) noisy 8×6 truss and (c) larger 9×6 truss. Results obtained using the same genotype without further re-optimization. Log-linear scale for cross-sectional areas.

	Opt.	Subopt.	Invalids
noisy	8	6	6
load 2×	20	0	0
fine grid	8	8	4

Tab 4.1. Controllers re-usability

	4×4	*8×6*
EiffelBlob	5.69 – 5.06	1.58 – 1.48
Direct encoding	4.98 – 4.92	1.50 – 1.46

Tab 4.2. Median and best ever of the best solutions weight$\times 10^{-3}$

truss structure to control beam size and length depending on beam strains. In this setup, a single cell controller is optimized and duplicated on all cells, which keeps the dimensions of the problem to the controller space. Experiments showed that comparable results can be achieved with *EiffelBlob* compared to a direct encoding approach. Even more important, resulting candidate solutions can be straight-forwardly reused for developping new and different trusses (different initial conditions and/or truss configuration), thus addressing to some extent issues of robustness and scaling, without having to pay the full price of restarting the optimization process.

So far, *EiffelBlob* has been applied with 2D trusses, however further extensions regarding 3D implementation mostly depends on the FEM simulation rather than the model itself. Ongoing work is also taking a deeper look into studying the reuse property, both from an experimental and theoretical viewpoints: this feature is indeed especially relevant when it comes to reusing a given candidate solution for much larger trusses, and re-developping several *EiffelBlob* models obtained on small trusses geometries can allow the engineer to quickly obtain one or several prototypes for very large trusses.

REFERENCES

[1] A. Auger and N. Hansen. A restart cma evolution strategy with increasing population size. In *CEC 2005*, pages 1769–1776, 2005.

[2] P. Bentley. Investigations into graceful degradation of evolutionary developmental software. *Natural Computing*, 4(4):417–437, 2005.

[3] J. C. Bongard and R. Pfeifer. Repeated structure and dissociation of genotypic and phenotypic complexity in artificial ontogeny. In *GECCO '01*, pages 829–836. Morgan Kaufmann, 2001.

[4] A. Devert, N. Bredeche, and M. Schoenauer. Robust multi-cellular developmental design. In *GECCO '07*, pages 982–989. ACM, 2007.

[5] A. Devert, N. Bredeche, and M. Schoenauer. Unsupervised learning of echo state networks: A case study in artificial embryogeny, 2007.

[6] P. Eggenberger. Evolving morphologies of simulated 3d organisms based on differential gene expression, 1997.

[7] N. S. Estévez and H. Lipson. Dynamical blueprints: exploiting levels of system-environment interaction. In *GECCO '07*, pages 238–244. ACM, 2007.

[8] D. Federici and T. Ziemke. Why are evolved developing organisms also fault-tolerant? In *SAB'06*, pages 449–460, 2006.

[9] F. Gruau. Genetic synthesis of modular neural networks. In *Proc. 5th ICGA*, pages 318–325. Morgan Kaufmann, 1993.

[10] T. Jakobsen. Advanced character physics. In *GDC 2001*, 2001.

[11] R. Kicinger, T. Arciszewski, and K. A. De Jong. *Morphogenic evolutionary design: cellular automata representations in topological structural design*, pages 25–38. Springer-Verlag, 2004.

[12] R. Kicinger, T. Arciszewski, and K. D. Jong. Evolutionary computation and structural design: A survey of the state-of-the-art. *Computers & Structures*, 83(23-24):1943–1978, 2005.

[13] R. Kicinger, T. Arciszewski, and K. D. Jong. Parameterized versus generative representations in structural design: an empirical comparison. In *GECCO '05*, pages 2007–2014. ACM, 2005.

[14] T. Kowaliw, P. Grogono, and N. Kharma. Environment as a spatial constraint on the growth of structural form. In *GECCO '07*, pages 1037–1044. ACM, 2007.

[15] J. F. Miller and W. Banzhaf. Evolving the program for a cell: from french flags to boolean circuits. In *On Growth, Form and Computers*. Academic Press, 2003.