

WHAT IS GENERATIVE IN GENERATIVE DESIGN TOOLS? UNCOVERING TOPOLOGICAL GENERATIVITY WITH A C-K MODEL OF EVOLUTIONARY ALGORITHMS

Armand ; Hatchuel, Pascal Le Masson, Maxime Thomas, Benoit Weil

▶ **To cite this version:**

Armand ; Hatchuel, Pascal Le Masson, Maxime Thomas, Benoit Weil. WHAT IS GENERATIVE IN GENERATIVE DESIGN TOOLS? UNCOVERING TOPOLOGICAL GENERATIVITY WITH A C-K MODEL OF EVOLUTIONARY ALGORITHMS. Proceedings of the Design Society: International Conference on Engineering Design, 2021. hal-03398565

HAL Id: hal-03398565

https://minesparis-psl.hal.science/hal-03398565

Submitted on 25 Oct 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

WHAT IS GENERATIVE IN GENERATIVE DESIGN TOOLS? UNCOVERING TOPOLOGICAL GENERATIVITY WITH A C-K MODEL OF EVOLUTIONARY ALGORITHMS.

Hatchuel, Armand;
Le Masson, Pascal;
Thomas, Maxime;
Weil, Benoit

MINES ParisTech-PSL

ABSTRACT

Generative design (GD) algorithms is a fast growing field. From the point of view of Design Science, this fast growth leads to wonder what exactly is 'generated' by GD algorithms and how? In the last decades, advances in design theory enabled to establish conditions and operators that characterize design generativity. Thus, it is now possible to study GD algorithms with the lenses of Design Science in order to reach a deeper and unified understanding of their generative techniques, their differences and, if possible, find new paths for improving their generativity.
In this paper, first, we rely on C-K ttheory to build a canonical model of GD, based independent of the field of application of the algorithm. This model shows that GD is generative if and only if it builds, not one single artefact, but a "topology of artefacts" that allows for design constructability, covering strategies, and functional comparability of designs. Second, we use the canonical model to compare four well documented and most advanced types of GD algorithms. From these cases, it appears that generating a topology enables the analyses of interdependences and the design of resilience.

Keywords: C-K theory, generative design algorithms, Design theory, Computational design methods, Design informatics

Contact:
Le Masson, Pascal
MINES ParisTech-PSL
Management Science
France
pascal.le_masson@mines-paristech.fr

Cite this article: Hatchuel, A., Le Masson, P., Thomas, M., Weil, B. (2021) 'What is Generative in Generative Design Tools? Uncovering Topological Generativity with a C-K Model of Evolutionary Algorithms.', in *Proceedings of the International Conference on Engineering Design (ICED21),* Gothenburg, Sweden, 16-20 August 2021. DOI:10.1017/pds.2021.603

1 INTRODUCTION: GENERATIVE DESIGN ALGORITHMS THROUGH THE LENSES OF DESIGN SCIENCE

Generative design algorithms (GDA) is a fast growing field that develops "design approaches that use algorithms to generate designs" (Caetano et al., 2020). Advances in CAD software (in particular with the launch of Autodesk so-called generative design (Kazi et al., 2017)), computing power and new computer science algorithms have contributed to the emergence of various "generative design tools" (Buonamici et al., 2020) with multiple applications. This fast growth leads to critical questions about generative design algorithms from the point of view of design science. What is exactly 'generated' and how? What means progress in the field? And more fundamentally, what can design science say for the theoretical understandings and enhancement of generative algorithms?

Design science have also achieved major steps. C-K theory has paved the way to new formalizations of a design process with a high level of generality. Design science is now able to establish conditions and operators that define and enable design generativity and generative power. Such notions are now independent of "what" is designed and of classic technical models. They describe generativity as the transformation of known objects into new ones using a new abstract language which received wide validation in the literature: concepts, generic extensions, restrictive and expansive partitions, identity of objects, independence and splitting structures of knowledge (Hatchuel et al., 2011, Le Masson et al., 2016). Thus, it is now possible to study Generative design algorithms with the lenses of Design Science in order to reach a deeper and unified understanding of their generative techniques, their differences and, if possible, find new paths for improving their generativity.

This paper develops such study through four steps that correspond to the sections of the article. In the literature review (part 2), we describe a brief state of the art of GDA and applications. In part 3, since any GDA is necessarily a constructive and iterative process, applying C-K theory to such constraints we can predict a general C-K logic for GDA. We define it as a *canonical model of GDA* that is not dependent of the field of application of the algorithm. This model mainly shows that GDA is generative if and only if it builds, not one single artefact, but a special "*topology of artefacts" that allows for design constructability, covering strategies, and functional comparability of designs*. In part 4, using the canonical model we compare the generative power of four well documented and most advanced types of GDA, which confirm their capacity to generate a 'topology of artefacts'.

2 LITERATURE REVIEW: THE VARIETY OF GD ALGORITHMS INVENTIONS

2.1 GD definition.

Recent reviews on GDA (Caetano et al., 2020, Mountstephens and Teo, 2020) share a definition of GDA: "a design approach that uses algorithms to generate designs" (Caetano et al., 2020); GDA is also seen "as the exploration of the principle of generating complex forms and patterns from a simple specification [with an algorithm]" (Shea et al., 2005, McCormack et al., 2005). Some authors insist on potential "creative outcomes" (Bernal et al., 2015) or "happy accidents" ie "unexpected results" born from "the number of design variations" and "the range of the variations", that positively impact the design process" (Chaszar and Joyce, 2016). Hence GDA definition focuses on the presence of an algorithm but remains relatively fuzzy on what exactly generativity consists in. The notion of the variety of designs provided by the software is key in GD A, even if the link between design variations and GD generativity is not clearly explained and loosely related to how GDA provokes 'surprises'.

2.2 A variety of Generative Design techniques.

GDA is clearly related to algorithmic rule-based processes that mainly refer to evolutionary techniques but are not limited to them (Caetano et al., 2020) (p. 294) - More specifically, (Mountstephens and Teo, 2020) identifies four generation methods: genetic algorithms, Shape grammars, L-Systems, Swarm intelligence - the authors mention other useful techniques: parametric modeling and topology optimisation. One could also add the recent use of techniques coming from AI like Generative Adversarial Networks (GAN) Variational AutoEncoders (VAE) to enhance topology optimization (Oh et al., 2019) or to enable shape parameterization for further generation processes (Burnap et al., 2016, Umetani, 2017). In this paper, we will rely on well-known Multi-Objective Generative Algorithms (also known as Multi-Objective Evolutionary Algorithms - MOEA) - and we will pay a particular attention to a new family of algorithms called quality-diversity, which are particularly relevant for GDA: quality-

diversity algorithms (Pugh et al., 2016) "evolve an archive of solutions which is, according to a user-defined behaviour space, as diverse as possible while obtaining for each solution a high performance" (Bossens et al., 2020). This family contains for the moment two prototypical algorithms: Novelty-Search with Local Competition (Lehman and Stanley, 2011a) and Multidimensional Archive of Phenotypic Elites (MAP-Elites) (Mouret and Clune, 2015).

2.3 Generativity in GD algorithms: a need for clarification and unification.

The field of GDA presents multiple streams of works that develop original algorithms applied to ad hoc design cases. Each software has its applications and illustrates one form of generativity. For instance:

- Byrne et al present a multi Objective Evolutionary Algorithm for design exploration and optimisation of a wing profile (Byrne et al., 2014);
- Multi-Objective Genetic Algorithm (in Autodesk) has been used in architectural space planning (Nagy et al., 2017), the design of office table (Nagy et al., 2017) (Chen et al., 2018);
- Novelty Search was used to evolve robot controllers into a deceptive maze (Pugh et al., 2016, Lehman and Stanley, 2011b) or to design images (Woolley and Stanley, 2011);
- Map-Eliteswas used to design a self-repairing hexapod robot (Cully et al., 2015) or wing profile (Cully et al., 2015) as well as in video game (Fontaine et al., 2019), automated image generation (Nguyen et al., 2015), robot morphologies and controllers (Hart et al., 2018);
- Autogenetic Design Theory was used for gearbox design (Wünsch et al., 2012).

This variety of applications calls for a clarification of the design logic associated to each GD software.

2.4 The locus of generativity in GDA: an engine to generate a population of artefacts?

Self-evidently, GDA operates on a parametrically defined object with constraints on the parameters, which restrict its use in design processes. (Mountstephens and Teo, 2020) proposed to distinguish between autonomous and interactive generative design: in a parametrically defined solution space, autonomous design might sound quite self-evident and generativity could appear only coming from interaction (capacity to use the GDA in a more or less creative way, at varied moments in the design process). Still, maybe counterintuitively, in this paper we *first* focus precisely on the specific generativity of the algorithm itself, its capacity to *generate a collection of varied artefacts*. Even if this generativity might sound limited ('parametric'?), we investigate how a unique property of GDA software lies in its capacity to offer a structured set of alternatives, and, *then*, how the user might react to this set.

In this paper, we focus on the generative logic of GDA such as: MOGA, quality-diversity algorithms, topologic optimisation, particle swarm optimisation, space filling techniques... Qualifying their generativity of these algorithms is not an easy task - and, to our knowledge, no systematic study was done until now. **Hence, our research question: how can we characterize formally and systematically the generativity logic of GDA 'engines'?**

3 A C-K CANONICAL MODEL TO UNCOVER THE GENERATIVITY OF GDA

3.1 Method: casting GDA in C-K theory.

C-K theory (Hatchuel and Weil, 2009), is one of the most advanced formulations of a design theory (Hatchuel et al., 2018). C-K theory presents the advantage to be independent of what is designed and can account for very strong forms of generativity (Hatchuel et al., 2011).

Casting a design method in C-K theory has already been done in other papers (Reich et al., 2012, Kroll et al., 2014). We will follow the same method. We first codify in a canonical model what can make the generativity of a GDA. Based on this analytical model, we analyse the generativity of a sample of published GDA.

3.2 A canonical model of GDA: concepts, object model, expansions.

3.2.1 Knowledge base - object model and splitting condition

A GDA operates on an initial state of knowledge K_0 which contains an object model M_0 with:

- Variables that can be assimilated to 'design parameters' or also called 'genotypical parameters' - parameters that are considered as directly actionable by the designer, noted $X_{i=1\ldots n}$.
- Variables that can be assimilated to functional (or behavioural or phenotypical) features, φ_j. They are not directly actionable; their value is computed in the object model: $X_{i=1\ldots n} \rightarrow \varphi_j$.
- One artefact is a point value $x_{i=1\ldots n}$ that has the features $\varphi_{j=1\ldots m}$.

GDA apply in situations where:

- The object model M_0 is *not invertible*: given specific $\varphi_{j=1\ldots m}$, the object model M_0 doesn't enable to find even one $X_{i=1\ldots n}$ that meets $\varphi_{j=1\ldots m}$ in finite reasonable time (see H1 below).
- The object model M_0 is not derivable not continuous, which means that a small change in one X_i can provoke strong changes in $\varphi_{j=1\ldots m}$ and conversely a small change in φ_j can correspond to a strong change in $X_{i=1\ldots n}$. In particular, this means that it is not possible, for a known object $(X_{i=1\ldots n}, \varphi_{j=1\ldots m})$, to know what is its *neighbourhood* in terms of genotype and (even less) in term of phenotype - so that in this kind of K_0, there is no self-evident solution to a problem of optimisation, ie finding an object that, at least locally, phenotypically dominates the others.

Design theory leads us to wonder whether this initial knowledge base K_0 is splitting (Le Masson et al., 2016), ie non-modular and non-deterministic. Modularity would mean that some design parameters could be added without effect on phenomenology. As we just mentioned, initially, in K_0, the design parameters are supposed to influence phenomenology. Determinism would mean that some design parameters would determine the phenomenological behaviour: again, as we just mentioned, initially, in K_0, one can't say that such deterministic law exist. So in usual contexts, we have:

> Property P1: usually *GDA operates on a knowledge base that meet the splitting condition, hence GDA is compatible with a generative process.*

3.2.2 Concepts as departures of a GDA:

Following C-K theory, any GDA that aims to design some X begins necessarily with a concept of the form "X that fulfils P(X)", P(X) being a series of properties of X such that:

- P(X) are undecidable in K_0 ie. there is no constructive rule that allows to design such X with K_0 (of course, since we want to describe the mechanism of the GDA 'engine' following the C-K operators, the 'engine' itself is not in K; otherwise it would appear as a constructive rule and the design is finished)
- P(X) will be constructible, true and compatible in some established K_n state of K.

It has to be underlined that here is a specific feature of GDA: GDA actually work to generate a *collection* of artefacts, ie in the concept {*X*, *P* (*X*)}, *X* actually refers to a collection of artefacts; and *P* actually refers to a property of this collection: in the concept "the set of wing profiles that form a Pareto front", we want to generate a collection (X) of wing profiles, and this collection has the property to form a Pareto front (this is a property of the population, and not of a lone artefact). Hence a second property:

> P2: usually GDA designs a *collection of artefacts* with specific property, this property can make that it is undecidable whether it is possible or impossible to get a population with property P.

P has to be interpretable (hence it is in K_0) and just needs to make X undecidable in K_0. Illustration:

- "a collection of N artefacts": it can be generated by instantiating M_0 N-times. It is not a concept.
- "a collection of N artefacts generated by random variation of genotype": it can easily be generated as soon as one knows of random number generator. This is not a concept.
- "a collection of N artefacts generated by variation of phenotype": if the object model is non-invertible, this is a concept.

3.2.3 Expansion in space K and concepts partition in space C

C-K theory models Concept partitions and expansions through tree structured sequences of nested partitions. They describe a constructive refinement of C_0 that should lead to the design of X; each of

these steps may activate space K, hence creating a knowledge expansion. At least, the last refinement produces an acceptable design that is integrated as a new true object in space K.
In the case of GDA, the algorithm is parameterized to produce knowledge and concept expansions. In a genetic algorithm, this is done by variation-selection. But the partitions can't be easily followed. When successive partitions can't be easily followed, it is possible to evaluate the expansions:

- If the initial proposition $\{X, P(X)\}$ was a concept and has become knowledge (in C-K terms: there was an initial disjunction and there is a final conjunction), then there was C-expansion.
- By comparing the knowledge base before (K_0) and after the GD process (K_{final}), one can estimate K-expansion.

P3: a criteria to evaluate the generativity of a GD is twofold: a- is there a conjunction after an initial disjunction? b- what is the knowledge expansion between K_0 and K_{final}?

P1, P2, P3 are the main properties of a canonical model of GDA in C-K framework (Fig. 2)

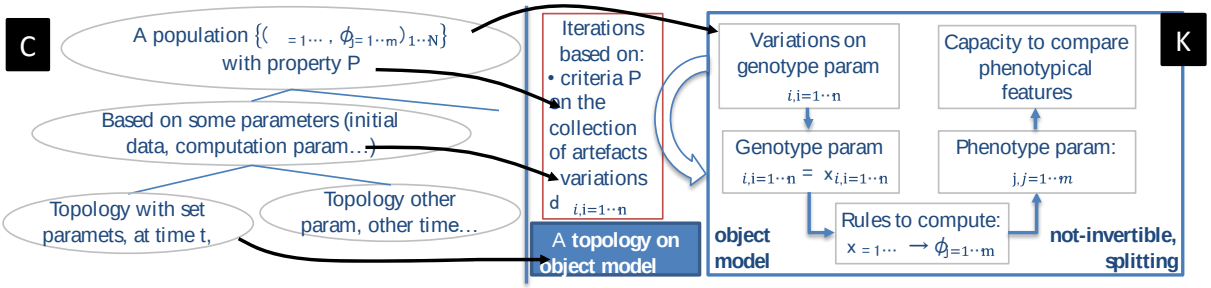


Figure 1. A canonical model of GD algorithms in C-K. P1: object model is non-invertible, a priori non continuous (hence splitting); P2: Property P is undecidable; P3: generativity is measured on disjunction-conjunction and K-expansions (a topology on object model M_0)

3.3 Avoiding the combinatorial trap in GDA: C-K conditions for generativity and the emergent topology of designs.

3.3.1 Variation and selection in GDA: the combinatorial trap

GDA raise a critical question for design theory: in which way can an algorithm be generative in the sense of C-K theory? Usual applications of C-K theory consider that K-space contains propositions (that are true or false) *as well as propositions that can be logically deducted one from the other* - hence not every new proposition is a K-expansion. Hence the knowledge base contains an internal "knowledge production engine" and one considers that there is expansion only if one goes 'beyond' this internal knowledge production engine. In the case of a GDA, our hypothesis is:

H1: we consider that classical computations techniques in finite time are available in K and their results are not considered as K-expansions. The algorithm that is under investigation is not in K.

3.3.2 Generativity in GD: the emergence of a topology of artefacts.

Building on properties P1 to P3 and hypothesis H1, what does C-K predict on GDA generativity?

- According to P1 and H1: even GDA-knowledge base is purely made of combinatorial knowledge, the knowledge base is splitting and enables generativity
- According to P2 and H1: concepts in GDA are related to specific properties associated to a collection of artefacts - the concept doesn't come from the number of entities (because of H1); hence the concepts comes from the structure and descriptors of this collection.
- According to P3: expansions can be evaluated by analysing initial and final C and final K-expansion. In the end of a GDA process, one gets a collection of artefacts that meet the structural property P. So that the generativity is in *this new structure of the collection of artefacts*.

At first sight, GDA appears trapped into a closed world of combinatorial designs. To avoid such trap, C-K theory calls for *thorough examination of all knowledge produced by the algorithm*. Clearly this knowledge is much more extended than the single artefacts that are designed. We have to recognize that the GDA not only explores single designs but compares them, positions them one to another, creating structures in the collection of artefacts. *GDA provides new knowledge on the topology of X_is.*

The expansion comes from the emergence of a 'geometry', a space in which artefacts can be relatively positioned. This new structure is a *topology on the model of objects, in the sense that:*

- GDA expresses each object in all its dimensions $\left(X_{i=1\ldots n}, \varphi_{j=1\ldots m}\right)$. Hence this space is multidimensional, linking genotypic dimensions and phenotypic dimensions.
- GDA enables to *distinguish* certain objects - each object of the final collection is carefully separated (in singletons).
- GDA also enables to *not* distinguish other objects: all the 'dominated' artefacts are considered in the same "neighbourhood"
- In this topology, the object model can be inverted (almost) everywhere: for each point of the topology, one relevant artefact can be associated (with respect to the criteria P). *It means that in the resulting topology the knowledge base is not splitting anymore.*

3.3.3 How topological knowledge provides a source of generativity for the user

For sure, there is a circular logic here: the topology that emerges is dependent of the iterative algorithm and another GDA technique would produce a different topology. Conversely, the information on the topology of designs can improve the GDA. However, what counts for the generativity of GDA is *the type of new knowledge* extracted from the topology that appeared. Information linked to this topology helps to explore dimensions of expansion predicted by canonical model Fig. 1:

- The topology can be extended by extending $X_{i=1\ldots n}, \varphi_{j=1\ldots m}$, and/or the model object M_0: adding or deleting some X_i, changing range, or modifying $\varphi_{j=1\ldots m}$.
- The topology of the $X_{i=1\ldots n}, \varphi_{j=1\ldots m}$ revealed by the population of designs can help to compute some property *P* that will be introduced to change the iteration rules. E.g an algorithm can use the density of designs in some areas of the X_is to evolve the selection rules.

4 GDA: UNCOVERING TOPOLOGIES AND COMPARING GENERATIVITY

We now have analytical tools (canonical model) and clear predictions (GDA tools generate topology of artefacts). We test them on a sample of most recent GDA. This sample was built on GDA recent reviews (Caetano et al., 2020, Mountstephens and Teo, 2020). We hence selected the following methods:

- MOEA (with one particular illustrative use case: (Byrne et al., 2014)),
- space-filling techniques (one particular illustrative use case: (Khan and Awan, 2018)),
- topological optimization (illustrative use case: (Matejka et al., 2018))
- Quality-Diversity (QD) algorithms (illustrative use case: (Clune et al., 2013))

4.1 Analysis of four GDA tools with the C-K canonical model

4.1.1 Multi-Objective Evolutionary Algorithm (MOAE)

Byrne et al. (Byrne et al., 2014) present a use case evolving parametric aircraft models. Coded with the canonical model (see Fig. 2 below): the designer knows a model of the aircraft, where given design parameters lead to two particular functional performances, Lift and Drag. The concept is: "A Pareto Front on the functions, max Lift, min Drag". The designer makes variations on a subset of three parameters among the set of possible design parameters and run the GDA tool, powered by a multi-objective evolutionary algorithm (MOEA) non-sorting genetic algorithm-II (NSGA-II) (Deb et al., 2002). This leads to a first Pareto front (see blue dots in the figure). Then the designer selects a larger set of parameters and run the algorithm again, to get another Pareto front (see red dots in the figure).

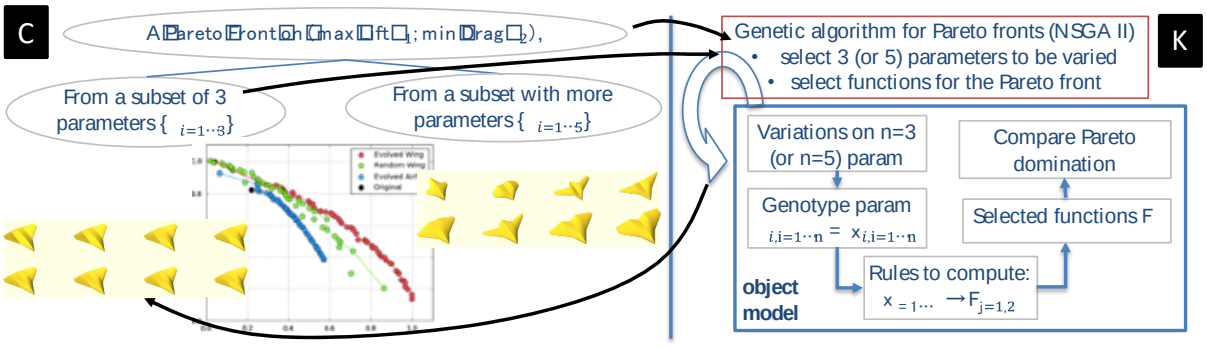


Figure 2. Pareto front GDA tool (MOEA NSGA-II), from Byrne et al 2014. Parametric object model in K. GDA generates the topology associated to a Pareto front.

Generativity analysed within the canonical model:

- the designer designs a Pareto front (not a single aircraft). We have a clear topology: single artefacts along the front, dominated artefacts below the front, no artefacts beyond.
- Modifying the design parameters to be varied, the designer gets several topologies (in case 1, the designer only evolved the wing profile, in case 2, the designer also evolved the fuselage).

4.1.2 Space-filling generative design

Khan & Awan (Khan and Awan, 2018) give (among others) an simple illustration of a "generative design technique for exploring shape variation", based on space-filling technique. In GD canonical model (see Fig. 3 below): in K, the designer disposes of a parameterized CAD-model (here a lamp, with two design parameters). The concept is: a map that represents the diversity of possible CAD shapes. To this end, the designer selects mapping criteria *P*: either space-filling (the criteria pushes to maximise the distance between shapes), or non-collapsing criteria (avoiding too different shapes (Draguljić et al., 2012)) or both. Powered also by MOES NSGA-II, the GD tool generates a map of CAD shapes.

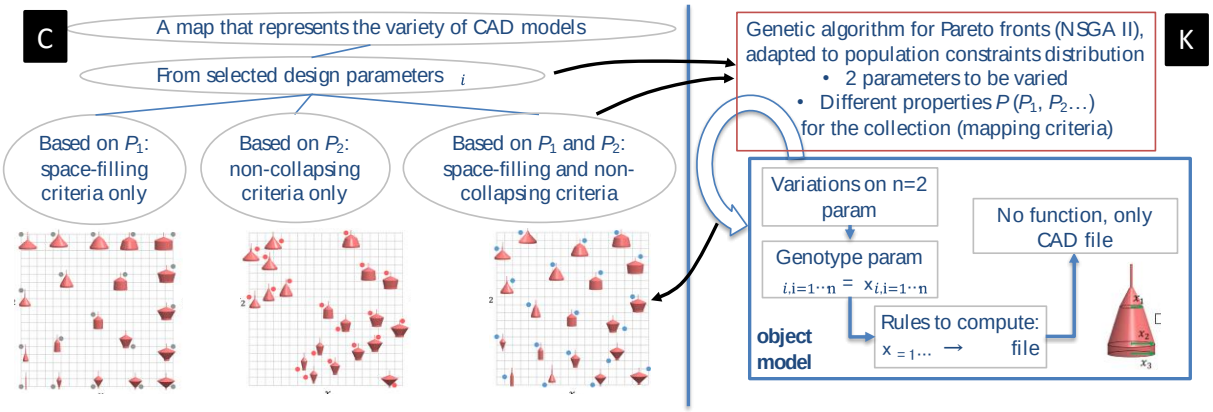


Figure 3. Space-filling GD tool (MOEA-NSGA-II) from Khan et al. 2014

Generativity analysed within the canonical model:

- the designer designs a map (not a single CAD shape) - hence a topology of artefacts.
- Formally speaking, the only difference with Pareto front case is that the former relies on mapping criteria whereas Pareto case, the mapping criteria are the function themselves.

4.1.3 Topological optimization algorithm

Matejka et al. present an example of topological optimization GDA for "exploration and visualization of large-scale generative design data set" (Matejka et al., 2018). In GD canonical model (see Fig. 4 below): in K, the designer has defined functions of the object (an office table) and the design parameters are only the presence or absence of matter. The concept can be formulated as "a large variety of (possibly surprising) office tables". To design one office table (CAD shape), the designer can fix a level to each of the constraints and use a topological optimization algorithm, optimizing the

office table weight while meeting the functional requirements - depending on optimization parameters (eg voxel size). By varying the functional levels and the optimization parameters, the designer gets a variety of shapes.

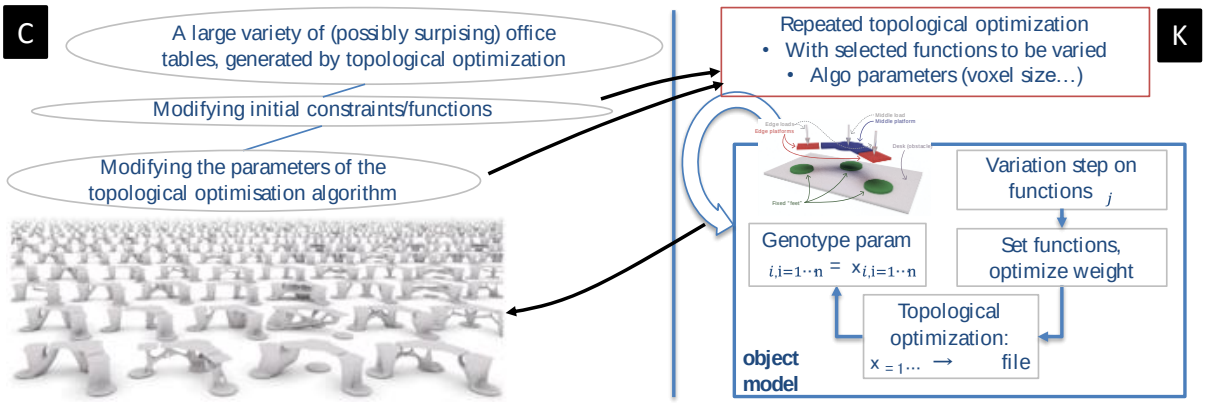


Figure 4. Topological algorithm GD tool coded in C-K - based on Matejka et al. 2018

Generativity analysed within the canonical model:

- the designer designs a variety of tables (X_i, F_j), on a functional map. Hence a topology.
- the object model is different: in MOEA case, design parameters determine functions; here the model can be punctually inverted: for one functional definition, topological optimization determines the design parameters (matter or not matter in each voxel). Still the model is only punctually invertible and so the topology is a concept.
- This GDA tool seems different from MOEA - but formally it leads to quite similar results: a set of artefacts ordered according to functional dimensions, with design parameters for each artefact.

4.1.4 MAP-Elites Quality-Diversity (QD) algorithm

MAP-Elites QD algorithm was used in a large variety of problems. One good illustration is the mapping of gaits of a hexapod robot (Cully et al., 2015, Koos et al., 2013). In GDA canonical model (see Fig. 5 below): the designer has a model of an hexapod described by 24 parameters, which create numerous gaits, from purely quadruped gait to classic tripod gait. The gait is a phenotype that can be described in several ways. For instance, one phenotype dimension can be φ_1=the speed in forward-axis - to be maximized. The concept can be formulated as: a map of gaits that present an optimal φ_1. This map is unknown and the designer can work *on the space in which* she will design these gaits. She has to propose a mapping criteria φ_2. She can map the φ_1-optimal gaits according to the y-axis speed - and this will probably result in a Pareto front. She can choose to map the gaits according to another phenotype dimension such as the fraction of time that each of the 6 legs touches ground (6-dimensional map). Based on φ_1 and φ_2, the MAP-Elites algorithm constructs an elite for each φ_2 niche.
Generativity analysed within the canonical model:

- The design process results in a topology of artefacts. Compared to Pareto front GDA, user relies on a phenotype feature that is not functional; compared to space-filling GDA, user defines his/her own phenomenological feature. *Hence mapping criteria is a free parameter of the tool*.
- The map can be seen as a generalization of topological optimization GDA tools: map dimensions are made by a phenomenological feature φ_2 and there is φ_1-optimized genotype (in topological space: for each set of functional levels, there is a shape that minimizes the weight).

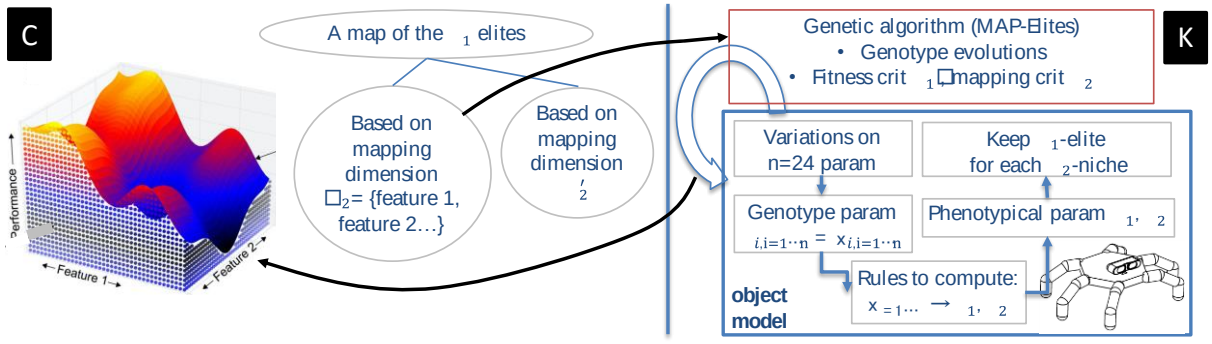


Figure 5. Map-Elitesalgorithm GD tool, from (Cully et al., 2015, Koos et al., 2013)

4.2 Characterizing the generativity of GDA tools: the design of topologies

We can synthesize the results just obtained above:

1. We confirm that GDA tools don't generate one artefact but generate **a topology of artefacts which is the locus of the generative power of the algorithm**.
2. In this topology design process, GDA tools differ (at least!) by **the degree of freedom they offer for building the topology.** Pareto front GDA tools maximize correlated functions; space filling techniques maximize phenotypic distances; topological optimization paves functional space with weight optimal artefacts; Map-Elites has the interesting (generic) property *to be free in term of phenomenological and genotypical features.*
3. As predicted, the GDA tools also open diverse directions for design expansions Each GDA tool generates a specific topology which impacts DP's or FR's generativity (see table below). One can predict the emergence of complementary GDA algorithms to work on the relevant $X_{i=1\ldots n}$, explore the $\varphi_{j=1\ldots m}$ and evolve the resulting M_0. (Gaier et al., 2017, Bossens et al., 2020). One can also predict the emergence of algorithms with double generativity (on DPs and FRs).
4. Analysing GDA as tools for designing topologies has several consequences for understanding their generativity, their value and their use:
   - **GDA tools help analyse interdependences between functions, not only correlation -as in Pareto front MOEA- but also independences - as in QD algorithm.**
   - The capacity to map independences can't be underestimated in engineering design: this is precisely the capacity required to design resilient systems, capable of being independent of external events. Hence **GDA tools might actually be useful for designing resilience.**
   - Mapping phenotypes to genotypical elites, GDA tools also contribute to "invert" usual models that compute functional requirements from design parameters. The maps generated by GDA tools gives a genotype for each phenotype niche. Doing so, GD A tool actually contributes to uncover general laws and models linking phenotypes to their genotypical roots. Hence it contributes to establish design rules. Mac Cormack (McCormack et al., 2005) considered that GDA tools would lead designers to focus on the design of design rules and GDA tools would then generate artefacts based on these new design rules. Fifteen years later, one could rather say that **GDA tools could be an essential tool to uncover design rules** and hence improves the generativity of users.

Table 1. Expansion directions opened by GD-generated topology

	Pareto front	Space filling	Topological optimization	MAP-Elites
Topology generativity	Pareto front topology (elites / dominated)	Identifies design neighbours	Cover functional map with (weight) optimal artefacts	Find elite gait for each specific behavioral niche
DP generativity	Identifies DP for improved Pareto front	None	May help identify new families of DPs (table legs…)	None
FR generativity	None	Paves the way to new FR, sensations, emotions	None	Explore altearntive behavioral niche types → resilient design

5 CONCLUSION

We analysed GDA in the light of design theory, to better characterize their generative power. Since these algorithms are based on a parametric of object, their generativity might sound limited to generate 'varied artefacts' from the same model, however we show that GDA design topologies of artefacts - ie large, ordered, structured, actionable sets of artefacts. GDA finally open the field of **topological generativity**. We show that GDA tools evolve today to enable more degree of freedom in the topology design, from the analysis of interdependences to the analysis of independences, leading possibly to designing resilience or to uncover design rules.

This design-theory based analyses of GDA tools helps uncover some of their critical properties, identifies some development trends and even suggests ways for further improvement. Conversely, the analysis invites to deepen design theory: with the help of C-K theory applied to Topos (Hatchuel et al., 2019) it might be possible to give an enriched account of the design of topologies by GDA tools, hence better addressing the issues of resilient design and the design of design rules.

REFERENCES

Bernal, M., Haymaker, J. R. & Eastman, C. 2015. On the role of computational support for designers in action. Design Studies, 41, 163-182.

Bossens, D., M, Mouret, J.-B. & Tarapore, D. Learning behaviour-performance maps with meta-evolution. GECCO'20 - Genetic and Evolutionary Computation Conf, 2020-07-08 2020 Cancun, Mexico.

Buonamici, F., Carfagni, M., Furferi, R., Governi, L. & Bvople, Y. 2020. Generative Design: An Explorative Study. Computer-Aided Design and Applications, 18, 144-155.

Burnap, A., Liu, Y., Pan, Y., Lee, H., Gonzalez, R. & Papalambros, P. Y. Estimating and Exploring the Product Form Design Space Using Deep Generative Models. ASME 2016 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference

Byrne, J., Cardiff, P., Brabazon, A. & O'Neill, M. 2014. Evolving parametric aircraft models for design exploration and optimisation. Neurocomputing, 142, 39-47.

Caetano, I., Santos, L. & Leitão, A. 2020. Computational design in architecture: Defining parametric, generative, and algorithmic design. Frontiers of Architectural Research, 9, 287-300.

Chaszar, A. & Joyce, S. C. 2016. Generating freedom: Questions of flexibility in digital design and architectural computation. International Journal of Architectural Computing, 14, 167-181.

Chen, X. A., Tao, Y., Wang, G., Kang, R., Grossman, T., Coros, S. & Hudson, S. E. 2018. Forte: User-Driven Generative Design. Proceedings of the 2018 CHI Conference Montreal QC, Canada: ACM.

Cully, A., Clune, J., Tarapore, D. & Mouret, J.-B. 2015. Robots that can adapt like animals. Nature, 521, 503-507.

Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6, 182-197.

Draguljić, D., Santner, T. J. & Dean, A. M. 2012. Noncollapsing Space-Filling Designs for Bounded Nonrectangular Regions. Technometrics, 54, 169-178.

Fontaine, M. C., Lee, S., Soros, L. B., Silva, F. D. M., Togelius, J. & Hoover, A. K. 2019. Mapping hearthstone deck spaces through MAP-elites with sliding boundaries. Proceedings of the Genetic and Evolutionary Computation Conference. Prague, Czech Republic: ACM

Gaier, A., Asteroth, A. & Mouret, J.-B. Data-Efficient Exploration, Optimization, and Modeling of Diverse Designs through Surrogate-Assisted Illumination (GECCO 2017), 2017 Berlin, Germany.

Hatchuel, A., Le Masson, P., Reich, Y. & Subrahmanian, E. 2018. Design theory: a foundation of a new paradigm for design science and engineering. Research in Engineering Design, 29, 5-21.

Hatchuel, A., Le Masson, P., Reich, Y. & Weil, B. A systematic approach of design theories using generativeness and robustness. ICED, 2011 Copenhagen, Technical University of Denmark. 12.

Hatchuel, A., Le Masson, P., Weil, B. & Carvajal Perez, D. Innovative design within tradition - injecting topos structures in C-K theory to model culinary creation heritage, 2019 Delft, Netherlands.

Hatchuel, A. & Weil, B. 2009. C-K design theory: an advanced formulation. Research in Engineering Design, 19, 181-192.

Kazi, R. H., Grossman, T., Cheong, H., Hashemi, A. & Fitzmaurice, G. 2017. DreamSketch: Early Stage 3D Design Explorations with Sketching and Generative Design. Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology. Québec City, QC, Canada.

Khan, S. & Awan, M. J. 2018. A generative design technique for exploring shape variations. Advanced Engineering Informatics, 38, 712-724.

Koos, S., Cully, A. & Mouret, J.-B. 2013. Fast Damage Recovery in Robotics with the T-Resilience Algorithm. ArXiv.

Kroll, E., Le Masson, P. & Weil, B. 2014. Steepest-first exploration with learning-based path evaluation. Research in Engineering Design, 25, 351-373.
Le Masson, P., Hatchuel, A. & Weil, B. 2016. Design Theory at Bauhaus: teaching “splitting” knowledge. Research in Engineering Design, 27, 91-115.
Lehman, J. & Stanley, K. 2011a. Evolving a diversity of creatures through novelty search and local competition, GECCO’11
Lehman, J. & Stanley, K. O. 2011b. Abandoning objectives: Evolution through the search for novelty alone. Evol. Comput., 19, 189-223.
Matejka, J., Glueck, M., Bradner, E., Hashemi, A., Grossman, T. & Fitzmaurice, G. 2018. Dream Lens: Exploration and Visualization of Large-Scale Generative Design Datasets. Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. Montreal QC, Canada: ACM.
Mccormack, J., Dorin, A. & Innocent, T. Generative design: a paradigm for design research. 2005.
Mountstephens, J. & Teo, J. 2020. Progress and Challenges in Generative Product Design: A Review of Systems. Computers, 9, 4.
Mouret, J.-B. & Clune, J. 2015. Illuminating search spaces by mapping elites. ArXiv.
Nagy, D., Lau, D., Locke, J., Stoddart, J., Villaggi, L., Wang, R., Zhao, D. & Benjamin, D. 2017. Project discover: an application of generative design for architectural space planning. Proceedings of the Symposium on Simulation for Architecture and Urban Design. Toronto, Canada:
Nguyen, A., Yosinski, J. & Clune, J. Innovation Engines: Automated Creativity and Improved Stochastic Optimization via Deep Learning. Proc. of the Genetic and Evolutionary Computation Conference. 2015.
Oh, S., Jung, Y., Kim, S., Lee, I. & Kang, N. 2019. Deep Generative Design: Integration of Topology Optimization and Generative Models. arXiv: Learning.
Pugh, J. K., Soros, L. B. & Stanley, K. 2016. Quality Diversity: A New Frontier for Evolutionary Computation. Frontiers Robotics AI, 3, 40.
Reich, Y., Hatchuel, A., Shai, O. & Subrahmanian, E. 2012. A Theoretical Analysis of Creativity Methods in Engineering Design: Casting ASIT within C-K Theory Journal of Eng. Design, 23, 137-158.
Shea, K., Aish, R. & Gourtovaia, M. 2005. Towards integrated performance-driven generative design tools. Automation in Construction, 14, 253-264.
Umetani, N. 2017. Exploring generative 3D shapes using autoencoder networks. SIGGRAPH Asia 2017 Technical Briefs. Bangkok, Thailand: Association for Computing Machinery.
Woolley, B. G. & Stanley, K. O. On the Deleterious Effects of A Priori Objectives on Evolution and Representation. Proceedings of the Genetic and Evolutionary Computation Conference., 2011.
Wünsch, A., Schabacker, M. & Vajna, S. 2012. Designing a gearbox for a novel independently controllable transmission using autogenetic design theory. 9th International Workshop on Integrated Product Development. Magdeburg, Germany.



CAMBRIDGE
UNIVERSITY PRESS