Towards Evolutionary Control Laws for Viability Problems

Alberto Tonda, Isabelle Alvarez, Sophie Martin, Giovanni Squillero, Evelyne Lutton

Towards Evolutionary Control Laws for Viability Problems

Alberto Tonda
UMR 518 MIA-PS, INRAE, Université Paris-Saclay, Palaiseau
Institut des Systèmes Complexes de Paris Île-de-France (ISC-PIF) - UAR 3611 CNRS, Paris
France
alberto.tonda@inrae.fr

Isabelle Alvarez
UMR 518 MIA-PS, INRAE, Université Paris-Saclay, Palaiseau
Institut des Systèmes Complexes de Paris Île-de-France (ISC-PIF) - UAR 3611 CNRS, Paris
France
isabelle.alvarez@inrae.fr

Sophie Martin
UMR 518 MIA-PS, INRAE, Université Paris-Saclay, Palaiseau
Institut des Systèmes Complexes de Paris Île-de-France (ISC-PIF) - UAR 3611 CNRS, Paris
France
sophie.martin@inrae.fr

Giovanni Squillero
DAUIN, Politecnico di Torino, Torino
Italy
giovanni.squillero@polito.it

Evelyne Lutton
UMR 518 MIA-PS, INRAE, Université Paris-Saclay, Palaiseau
Institut des Systèmes Complexes de Paris Île-de-France (ISC-PIF) - UAR 3611 CNRS, Paris
France
evelyne.lutton@inrae.fr

ABSTRACT

The mathematical theory of viability, developed to formalize problems related to natural and social phenomena, investigates the evolution of dynamical systems under constraints. A main objective of this theory is to design control laws to keep systems inside viable domains. Control laws are traditionally defined as rules, based on the current position in the state space with respect to the boundaries of the viability kernel. However, finding these boundaries is a computationally expensive procedure, feasible only for trivial systems. We propose an approach based on Genetic Programming (GP) to discover control laws for viability problems in analytic form. Such laws could keep a system viable without the need of computing its viability kernel, facilitate communication with stakeholders, and improve explainability. A candidate set of control rules is encoded as GP trees describing equations. Evaluation is noisy, due to stochastic sampling: initial conditions are randomly drawn from the state space of the problem, and for each, a system of differential equations describing the system is solved, creating a trajectory. Candidate control laws are rewarded for keeping viable as many trajectories as possible, for as long as possible. The proposed approach is evaluated on established benchmarks for viability and delivers promising results.

CCS CONCEPTS

• **Computing methodologies** → *Continuous space search*; • **Theory of computation** → **Genetic programming**.

GECCO '23, July 15–19, 2023, Lisbon, Portugal

ACM ISBN 979-8-4007-0119-1/23/07…$15.00
https://doi.org/10.1145/3583131.3590415

KEYWORDS

genetic programming, machine learning, viability theory, viable feedback

ACM Reference Format:
Alberto Tonda, Isabelle Alvarez, Sophie Martin, Giovanni Squillero, and Evelyne Lutton. 2023. Towards Evolutionary Control Laws for Viability Problems. In *Genetic and Evolutionary Computation Conference (GECCO '23), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3583131.3590415

1 INTRODUCTION

The viability approach is a theory that describes the properties of a dynamical system as a map on its state space. Each state can be characterized according to the dynamics of the system with respect to a set of constraints. Important notions such as resilience have been proposed [27], that allowed revisiting multi-criteria problems, in particular in the field of ecology, economy and sustainable development (for a review, see [34]), planetary boundaries [29], chemistry [41], and other fields. The main idea is to provide a different way to consider optimality: instead of searching an "optimal" solution, a set of solutions satisfying some constraints is searched. These constraints can reflect complex issues, like for instance conciliation between stakeholders [2].

A major concern in this field is related to the computational costs of the algorithms, they are often very high. Once a viability problem is formulated, a standard approach is to identify the viability kernel that gathers all the states from which it is possible to maintain at least one trajectory in the constraint set. Exact analytic solutions are very rare: for examples in 2D see [5], Chapter 7; and for an example in 3D, see [7]. Computation of approximations is difficult and time-consuming since general algorithms are exponential with the dimension [37]. And more computation is generally necessary to obtain the map of viable controls. Only a few general control laws are available, for instance the slow or quasi-slow selections of viable controls which select the viable controls with minimal norm

[11], or heavy strategy which selects viable controls whose velocity has at each instant the minimal norm [6]. Unfortunately these strategies often lead trajectories to the boundary of the viability kernel, which is not desirable, since outside the viability kernel trajectories are doomed to leave the constraint set. Heuristic prudent trajectories [1] can be considered to stay far from the boundary, but the computation of trajectories robust to perturbation is very time consuming [28].

The original idea of this paper is to use genetic programming (GP) to compute directly a viable control map in analytic form. The idea of using evolutionary computation for finding a viable trajectory once the viability kernel has been computed is not new: [30] proposed an approach applied to a complex agrifood process. The novel idea proposed in this work is to employ the GP engine to automatically discover a control law in analytic form, a set of equations encoded as GP trees, removing the need of exactly computing the boundaries of the viability kernel. The analytic control law to be discovered is tasked with keeping as many trajectories as possible inside the problem constraints: a candidate control law is considered to be better than another if it is able to keep a larger number of trajectories viable for longer. An experimental evaluation on two known viability benchmarks show that the approach is able to deliver encouraging results.

The paper is organized as follows: in section 2 we briefly present the viability theory, and, as the evaluation of a control relies on a stochastic sampling of the state space, we recall some issues related to noise management in evolutionary computation and genetic programming. Section 3 describes the proposed approach and details the structure of an individual, the evolutionary operators and the fitness evaluation procedure based on a stochastic sampling. An experimental analysis is performed on two classical use cases of the domain in 2 and 3 dimensions (section 4), showing promising results. We finally draw some conclusions in section 5 and discuss about future extensions of this work.

2 BACKGROUND

In the following we outline the basic concepts of viability theory and genetic programming, necessary to introduce the scope of our work.

2.1 Viability theory

Mathematical viability theory, as defined by [3], is a mathematical framework designed to study the compatibility between evolutions of dynamical systems and constraints in the state space. A viability problem is generally defined with a controlled dynamical system S, which may be subject to uncertainties, called "tyches" in the formal definition below[1], a set of admissible controls U, which may depend on the state of the system, and a set of constraints, K, which is a subset of the state space. The viability kernel $Viab_S(K)$ is a subset of the constraints set K that gathers all the states from which it is possible to maintain at least one trajectory within K. In practice, most viability problems can be defined as a system of Ordinary Differential Equations (ODE).

[1] "Tyche was the presiding tutelary deity who governed the fortune and prosperity of a city, its destiny." from https://en.wikipedia.org/wiki/Tyche

For example, for a problem of lake eutrophication (detailed in Section 4.1), when human activities can lead to water pollution, the hysteresis dynamics of the lake is well described by the evolution of variables L, the input of phosphorus, and P, the concentration of phosphorus in the lake [27]. The constraint set defines an upper limit for P to prevent eutrophication, and a lower limit for L to support human activities. Figure 1 shows the viability kernel for this case study, using the same parameters described in Section 4.1.

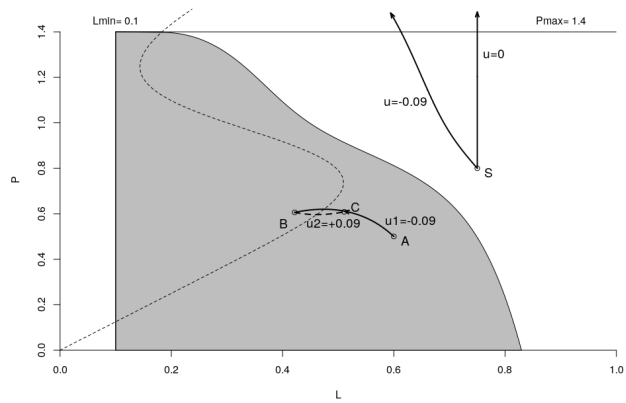

Figure 1: Viability kernel of the lake eutrophication case study (Eq. 10) in gray color. The dotted line shows the equilibria of the dynamics ($\frac{dP}{dt} = 0$). State A is in the viability kernel, we can see a trajectory starting at A and staying in the constraint set forever, with control $u1 = -0.09$ from A to B, then control $u2 = +0.09$ from B to C, and then alternatively control switch from $u2$ to $u1$ at state C and from $u1$ to $u2$ at state B. State S is in the constraint set but outside the viability kernel: All trajectories starting at S leave the constraint set in finite time, no matter the value of u.

Formally, a viability problem is defined as follows. Consider the set of variables that govern the behaviour of a system, a classical guaranteed viability kernel mathematical formulation [3, 4] is:

- Let $x \in R^n$ be the state vector, that describes the current state of the system.
- Let $u \in R^p$ be the control vector, that represents the variables that can be changed for controlling the system.
- Let $v \in R^q$ be the so-called tyche vector, that represents all variables that vary but cannot be controlled.

The map

$$U : R^n \rightsquigarrow R^p \tag{1}$$

defines the set of admissible controls, the map

$$V : R^n \rightsquigarrow R^q \tag{2}$$

defines the set of anticipated tyches and the function

$$f : R^n \times R^p \times R^q \rightarrow R^n \tag{3}$$

describes the evolution of x over time

$$\begin{cases} x(t+1) = f(x(t), u(t), v(t)) & \text{state variables} \\ u(t) \in U(x(t)) & \text{controls} \\ v(t) \in V(x(t)) & \text{tyches} \end{cases} \tag{4}$$

Finally, the constraints set on the system are described by a set

$$K \subset R^n \tag{5}$$

The answer to the viability problem between f, U and K is a map

$$\tilde{R} : R^n \rightsquigarrow R^p \tag{6}$$

such that all evolutions starting in $Dom(\tilde{R})$ and governed by $f, \tilde{R}, V$ remains in K forever. The biggest map for the inclusion of its graph is called the regulation map and its domain $Viab_S(K) = Dom(\tilde{R})$ is called the guaranteed viability kernel, or simply the viability kernel when tyches are not considered. Finding the viability kernel is computationally expensive, as current techniques require a recursive elimination of non-viable states, for which no viable controls can be found; and for most real-world practical applications, the exact computation is impossible.

In this work, rather than approximating the viability kernel from the constraint state, we propose to address viability problems by searching directly for a control law encoded as an analytic function $U(x)$ of the current state x. Search for an analytic function $U(x)$ can be performed through an approach based on Genetic Programming.

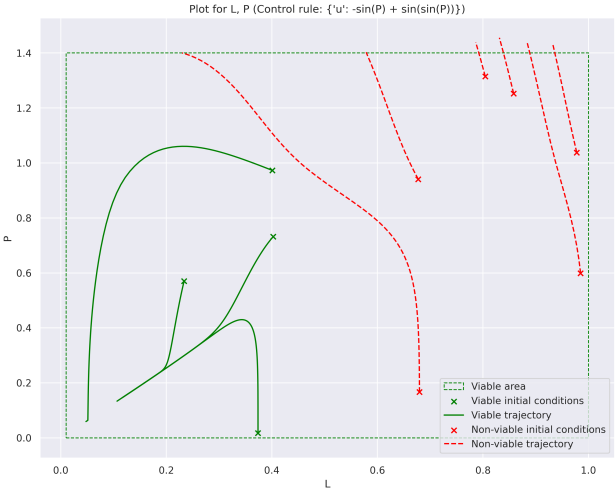

Figure 2: Example of viable and non-viable trajectories, given the viability problem described by Eq. 10 and an analytic control law. The state variables of the problem are L and P, and the control law $u(L, P) = -\sin(P) + \sin(\sin(P))$ is able to successfully keep some of the trajectories (in green) inside the viable area of the given constraints, while other trajectories (red, dotted lines) eventually escape the viable area.

2.2 Genetic programming with noisy fitness

Genetic Programming (GP) [21] is an established evolutionary optimization technique, able to optimize trees describing programs or functions. GP can boast several successful applications [13, 19], especially for symbolic regression [39], where GP trees encode equations.

Using GP for viability problems puts us in a context where the behaviour of a dynamical system has to be assessed over a whole continuous domain. This evaluation is usually accessible through stochastic sampling, spatially (for the domain) as well as within a finite temporal horizon (simulations with a finite number of steps). As a consequence, the associated fitness function to be optimized is often very noisy.

Since a long time, evolutionary computation has been proved to be able to deal rather smoothly with noisy landscapes due to their intrinsic stochastic nature [12, 14, 15, 20]. Theoretical analyses [10, 36] have shown that noise has no impact on the limit distribution (i.e. the position of the global optimum of the fitness function), but on the way it is reached, evidencing the importance of finite population effects. Noisy fitness functions may necessitate longer computations to reach a reliable optimum. Other analyses have provided recommendations on population sizing [14], selection methods [32] or sampling strategies [8]. The efficiency of sampling strategies (i.e. using repeated fitness evaluations to reduce the noise) remains however an issue: it has been shown that in some conditions, a "classical" strategy (simply ignoring noise) is able to outperform repetition-based algorithms [22]. Repeated fitness computations have a non negligible computational cost, that must be finely balanced with the other computations of the evolutionary search process. Adaptive [8] or memory-based [38] strategies have been proposed in order to reduce the computation time in terms of function evaluations.

The impact of noise in GP has been less studied. Recent works [18, 26] have however shown that it has also a noticeable impact on the performances of the algorithms, such as GP-based applications for machine-learning (generalization capabilities) [33], classification [40] or symbolic regression [17].

Surrogates, or approximate fitness functions (usually considered when the full computation is expensive), proposed at the end of the last century [35] may provide another interesting viewpoint on the question. They have been lately used in GP [18], but seem interesting to accelerate computations and improve quality of results. In this context very rough approximations of the fitness functions can be used – often computed by means of statistical or machine learning from samples of previous evaluations – then progressively refined with respect to the state of the evolution and/or the value of the estimated fitness. These algorithms based on surrogates share common features with algorithms dealing with noisy fitness. They actually highlight an interesting characteristic of progressive approximations of fitness calculations: besides the straightforward computational advantage of shorter unnecessary calculations, it seems that such strategies provide smoother landscapes to the evolutionary process, thus facilitate its convergence. This feature has been theoretically investigated in canonical EAs, see for instance [23, 24, 25] for studies on the influence of fitness regularity; the extension of this analysis to GP landscapes is an open and challenging question.

3 PROPOSED APPROACH

In this work, we propose a novel approach for tackling viability problems, based on the idea of directly searching for analytic control laws U, represented as a set of GP-trees, where functions can be min-max bounded to fit the limits of the domain of control values. Each control is evaluated thanks to an estimation of its ability to

generate trajectories that do not escape from the set K. As we are dealing with a continuous region, the estimation is performed via a random sampling of initial conditions inside K. We thus have to deal with a noisy fitness, corresponding to a stochastic approximation of the real fitness value. To limit overfitting, n_{ic} initial conditions are randomly chosen at each generation, and all individuals of the same generation are compared on the same set of initial conditions.

3.1 Structure of a candidate solution

Each viability problem has a given number of control laws, which make it possible to influence the values of its state variables, and possibly keep them within constraints, in a viable state. We propose to model a candidate set of control laws for a given viability problem as a set of equations:

$$u(x_1, ..., x_n) \begin{cases} u_1 = G_1(x_1, ..., x_n) \\ u_2 = G_2(x_1, ..., x_n) \\ ... \\ u_n = G_n(x_1, ..., x_n) \end{cases} \tag{7}$$

where $u = (u_1, ..., u_n)$ is the control law for a given viability problem, $x = (x_1, ..., x_n)$ is the vector of state variables of the viability problem, and $G_1, ..., G_n$ are different GP trees representing equations, with function set and terminals typical of classical symbolic regression approaches [21, 39].

3.2 Evolutionary operators

Every time an individual is selected for reproduction, for each control law inside the individual, an evolutionary operator is randomly chosen to create a new law. The considered operators are classical for GP applications: one-point crossover, applied with probability p_c; hoist mutation, with probability p_h; point mutation, with probability p_p; subtree mutation, with probability p_s; and a final replication operator that simply copies the parent's GP tree, applied with probability $p_r = 1.0 - (p_c + p_h + p_p + p_s)$. If the one-point crossover is chosen, a second individual is selected and the crossover is performed with the corresponding control law of the second individual. If a child individual is left identical to the parent, due to the replication operator being selected for each control law, the child individual is discarded and the process is iterated until a child individual with at least one different law is eventually produced.

3.3 Fitness evaluation using stochastic approximation

Evaluating the goodness of a candidate set of control laws for a given viability problem seems straightforward: given random initial conditions for the ODE system describing the problem, solve the system using the candidate control laws for an integration step Δ_t and maximum time $\max_t$, and finally check whether the resulting trajectory stays inside the boundaries of the given constraints until the end. This evaluation, however, is not appropriate for a fitness function, as (i) it only provides a yes/no (viable/non-viable) answer, which is too coarse for evolutionary optimization methods and (ii) it neglects the consideration that some initial conditions may be non-viable, no matter what set of control laws is applied.

In order to smoothen the fitness function, we thus proceed to: (i) draw a user-defined number of random initial conditions n_{ic} from the state space uniformly within the constraints at each generation, and evaluate all individuals of the same generation on the same set of initial conditions; (ii) assign a reward to the candidate solution equal to the number of viable trajectories that stay inside the boundaries defined by the constraints for the whole $\max_t/\Delta_t$ time steps; and (iii) even for non-viable trajectories, we assign a partial reward proportional to the number of time steps for which the trajectory managed to stay within the boundaries. Using another formulation, the fitness function for candidate solution I, given n_{ic} initial conditions is:

$$F(I) = \frac{\sum_{i=1}^{n_{ic}} r(i)}{n_{ic}} \tag{8}$$

with reward function $r(i)$ being:

$$r(i) = \begin{cases} 1.0 & \text{if trajectory for initial conditions } i \text{ is viable} \\ \frac{v_i}{\max_t/\Delta_t} & \text{otherwise} \end{cases} \tag{9}$$

where v_i is the number of time steps for which trajectory i remained viable, and $\max_t/\Delta_t$ gives the total number of time steps. In other words, a non-viable trajectory j that violates the constraints on the first time step (out of 100) will have a reward $r(j) = 0.01$, while a non-viable trajectory k that stays inside the boundaries for 99 steps will have a reward $r(k) = 0.99$. This could provide enough information to the evolutionary algorithm to be able to separate the performance of different candidate solutions. Taking for example the candidate solution for the viability problem depicted in Figure 2, the individual will receive a full reward for the viable trajectories in green, and partial rewards for the ones in red, with the biggest partial reward for the trajectory starting from the bottom right part of the state space, as it stays inside the constraints for longer.

Fitness function F will thus range between 0 and 1, and is to be maximized. It is interesting to notice that, depending on a viability problem's characteristics, the fitness function might never reach its theoretical maximum value, as trajectories for some initial conditions will never be viable, independently from the control selected: only trajectories starting inside the viability kernel (that is in principle unknown and expensive to compute) have a chance of staying inside the constraints, given the right control. It is also relevant to remark that, as the initial conditions for each generation are randomly generated, the fitness function can be characterized as noisy.

4 EXPERIMENTAL EVALUATION

To validate the proposed approach, we select two known benchmarks in viability theory, an ODE system with two equations, describing lake eutrophication, with one control law; and a ODE system with three equations, describing a sphere, featuring three control laws. For all considered benchmarks, we do not add tyches, to simplify the problems for an initial assessment of our approach, and to reduce possible sources of stochasticity that would increase the noise in the fitness evaluation.

The proposed approach is implemented in Python 3.9, employing libraries `sympy` [31] for symbolic computation, `scipy` [42] for

integration of ODE systems, and classes from `gplearn`[2] for the GP engine at the core of the proposed approach. All the necessary code to reproduce the experiments and figures can be found on the GitHub repository: https://github.com/albertotonda/evolutionary-viability-theory.

After preliminary experiments, the parameters of the employed algorithms are set as follows, for all the considered case studies. The function set for the GP-evolved control laws consists of:

$$\{+, -, *, /, \log, \sqrt{}, \sin, \cos\}$$

and the terminals include all experiment-specific state variables, plus constant floating point values $c \in (-1.0, 1.0)$. The initialization method for the starting population is *half and half*, with 50% of the trees grown choosing at random functions or terminals for each node, and the remaining 50% grown choosing random functions from the function set until the maximum allowed depth is reached, and then choosing random terminals. The GP trees in the initial population will have a depth $2 \leq d \leq 4$. The activation probabilities of the genetic operators are set as: $p_c = 0.4, p_h = 0.1, p_s = 0.1, p_p = 0.1, p_i = 0.3$. The selection for reproduction is performed using a tournament selection of size $\tau = 2$. The evolutionary algorithm employs a replacement strategy $(\mu + \lambda)$.

All experiments are run on a server with an AMD EPYC 7301 16-Core Processor and 128 GB of RAM. Fitness evaluations of candidate individuals have been parallelized using the multi-process options of `inspyred`.

4.1 Case study: lake eutrophication

4.1.1 Problem description. Eutrophication is a phenomenon affecting bodies of water, by which they become progressively enriched with minerals and nutrients, particularly nitrogen and phosphorus, increasing the growth of phytoplankton. When this process starts rapidly affecting water that is normally oligotrophic, thus very low in nitrogen and phosphorus, the effects can be extremely deleterious on the lake's ecosystem. Several lakes have experienced sudden shifts from an oligotrophic to an eutrophic state. This phenomenon is due to excess phosphorus in the lake. The model describes the dynamics of phosphorus concentration and phosphorus inputs over time. The issue is to determine whether it is possible to maintain the lake in an oligotrophic state while preserving agricultural activities in its watershed. For this particular application, it is possible to act on the input of phosphorus in the lake, as a residual of agricultural activities. This is a classical viability problem, whose description is available from the ViNO platform[3].

From [9], we consider that the phosporus concentration dynamics is governed by :

$$\begin{cases} L(t+1) = L(t) + u \\ P(t+1) = P(t) - b \cdot P(t) + L(t) + r \cdot \frac{P(t)^q}{P(t)^q + m^q} \end{cases} \quad \text{with } u \in [u_{min}, u_{max}] \tag{10}$$

where L is the input of phosphorus, P is the concentration of phosphorus, u is the variation in the input of phosphorus, and b, r, m, q, u_{min} and u_{max} are parameters.

[2]https://gplearn.readthedocs.io/en/stable/intro.html
[3]https://lisc.inrae.fr/vino/

4.1.2 Experimental setup. For the simulations below, we used the `2D-lake-PSP-16384ppa` benchmark [4] with the following parameters:

$$b = 0.8, r = 1.0, q = 8.0, m = 1.0$$

and the following constraints:

$$L_{min} = 0.1, L_{max} = 1.0, P_{min} = 0.0,$$
$$P_{max} = 1.4, u_{min} = -0.09, u_{max} = 0.09$$

This particular case study includes a single control law, u, that we define as an equation $u(L, P)$ to be found through the proposed approach. As the problem includes a constraint $u_{min} \leq u(L, P) \leq u_{max}$, we define a candidate solution as:

$$u(L, P) = \begin{cases} u_{min} & \text{if } u < u_{min} \\ G(L, P) & \text{if } u_{min} \leq u \leq u_{max} \\ u_{max} & \text{if } u > u_{max} \end{cases} \tag{11}$$

where $G(L, P)$ is the equation represented by a GP tree. We decide to impose these threshold to prevent the evolutionary approach from employing computational time to autonomously rediscover a way to respect this constraint. In practice, this corresponds to setting $u(L, P) = \max(u_{min}, \min(u_{max}, G(L, P)))$.

The experimental run uses the following parameters for the evolutionary algorithm: $\mu = 100, \lambda = 200$ and a termination condition set on the maximum number of evaluations, $E_{max} = 10,000$. Each candidate solution is evaluated on $n_{ic} = 100$ randomly selected initial conditions. For the integration of the ODE system representing the problem, the solver selected is an explicit Runge-Kutta method of order 4 [16], with $\Delta_t = 0.01$ and $\max_t = 100$. The experimental run lasted a total of 30 hours on the previously described hardware configuration.

4.1.3 Results. Analyzing the results, the fitness value of the best individual grows over the generations as expected for evolutionary computation approaches; but since the fitness evaluation is noisy, with a new set of initial conditions randomly generated at each iteration, a further validation is needed to assess the improvements. Figure 3 shows a comparison between the best individual at generation 0 and the best individual in the final generation, using 500 randomly generated initial conditions, different from those seen during the evolutionary process. The area highlighted in blue in the figure represents the viability kernel, that for this particular benchmark is known. Remarkably, the best control law found in the experiment keeps all trajectories that originate inside the viability kernel viable. With reference to Eq. 11, the best control found at the end of the experiment has the form:

$$G(L, P) = \sin(\log(L)) - (\log(L) - 1.4617) \tag{12}$$

4.2 Case study: 3D-sphere

4.2.1 Problem description. This case study is another classical viability benchmark, again found in the ViNO platform, with the ODE system describing a sphere in three dimensions:

[4]Available at https://demo.vino.openmole.org/viabilityproblem/1/#kernel/1/ppa/1000/

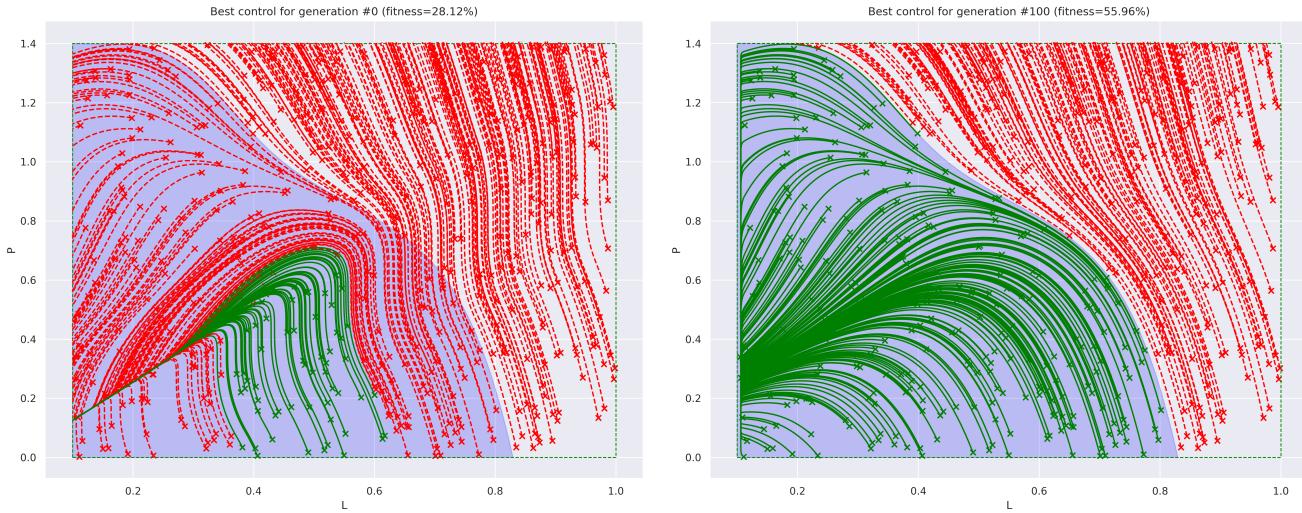

Figure 3: Analysis of individuals for the lake eutrophication case study, using the same 500 randomly generated initial conditions, not seen during the evolutionary optimization. The fitness value is reported as a percentage of the total number of initial conditions. The area in blue represents the viability kernel, that is known for this particular benchmark [9]. (Left) best individual at generation 0, (Right) best individual at generation 100.

$$\begin{cases} x(t+1) = x(t) + a \cdot u_x \\ y(t+1) = y(t) + a \cdot u_y \\ z(t+1) = z(t) + a \cdot u_z \end{cases} \quad \text{with } u_x^2 + u_y^2 + u_z^2 \leq 1 \tag{13}$$

where u_x, u_y, u_z are the three control laws, and the viable volume in the state space is constrained by:

$$x^2 + y^2 + z^2 \leq r \tag{14}$$

4.2.2 Experimental setup. For this experiment, we consider the following parameters, taken from problem instance `3D-ball-Viablab-81ppa`[5]: $a = 1.0, r = 1.5$.

The experimental run uses the following parameters for the evolutionary algorithm: $\mu = 100, \lambda = 200$ and a termination condition set on the maximum number of evaluations, $E_{max} = 10,000$. Each candidate solution is evaluated on $n_{ic} = 100$ randomly selected initial conditions. For the integration of the ODE system representing the problem, the solver selected is an explicit Runge-Kutta method of order 4 [16], with $\Delta_t = 0.01$ and $\max_t = 100$. The experimental run lasted a total of 90 hours on the previously described hardware configuration.

4.2.3 Results. The 3D-sphere benchmark proves to be considerably more challenging than the lake eutrophication case study. The best fitness value in generation 0 is 0.0016, far below 0.01, indicating that a randomly generated control law typically cannot keep even one single trajectory inside the viable volume described by the constraints.

Figure 4 shows that a considerable number of non-viable trajectories stop before crossing the constraint in the state space: this indicates that they violate the constraint imposed on the control laws, $u_x^2+u_y^2+u_z^2 < 1.0$. Apparently, for this case study, the proposed approach has difficulty discovering a way around this obstacle: even after 100 generations, the best individual still has fitness 0.0061, indicating that the candidate control law managed to keep the trajectories viable for longer, but in the end they all escaped the boundaries of the constraints.

These results are not completely unexpected, as in this case setting thresholds for each control equation separately is not possible; and the search space to be explored by the proposed GP-based approach is substantially larger, as three GP trees need to be optimized, instead of just one. With reference to Eq. 13, the best candidate control law found is:

$$\begin{cases} u_x(x,y,z) = y \cdot \frac{-0.9655 \cdot x}{y} \\ u_y(x,y,z) = \cos(\cos((y + \log(\cos(-0.4505 \cdot z))) \cdot 0.3767)) \\ u_z(x,y,z) = z \cdot \sin(-0.6640) \end{cases} \tag{15}$$

It could be possible to frame the same problem in spherical coordinates, which might make the evolutionary search more efficient, as in a spherical coordinates system it could be easier to express the constraints of Eq. 13 and Eq. 14 as separate constraints on each control rule, and replicate the individual encoding used in the lake eutrophication case study, described by Eq. 11. Further experiments are needed to explore this possibility.

5 CONCLUSION AND PERSPECTIVES

In this work, we presented a novel evolutionary approach to the automatic discovery of analytic control laws for viability problems. Viability theory traditionally approached control laws by employing rules or by performing local optimization after each integration step of an ODE system representing the viability problem. To the best of the authors' knowledge, this is the first time that the problem

[5]https://demo.vino.openmole.org/viabilityproblem/6/#kernel/12/

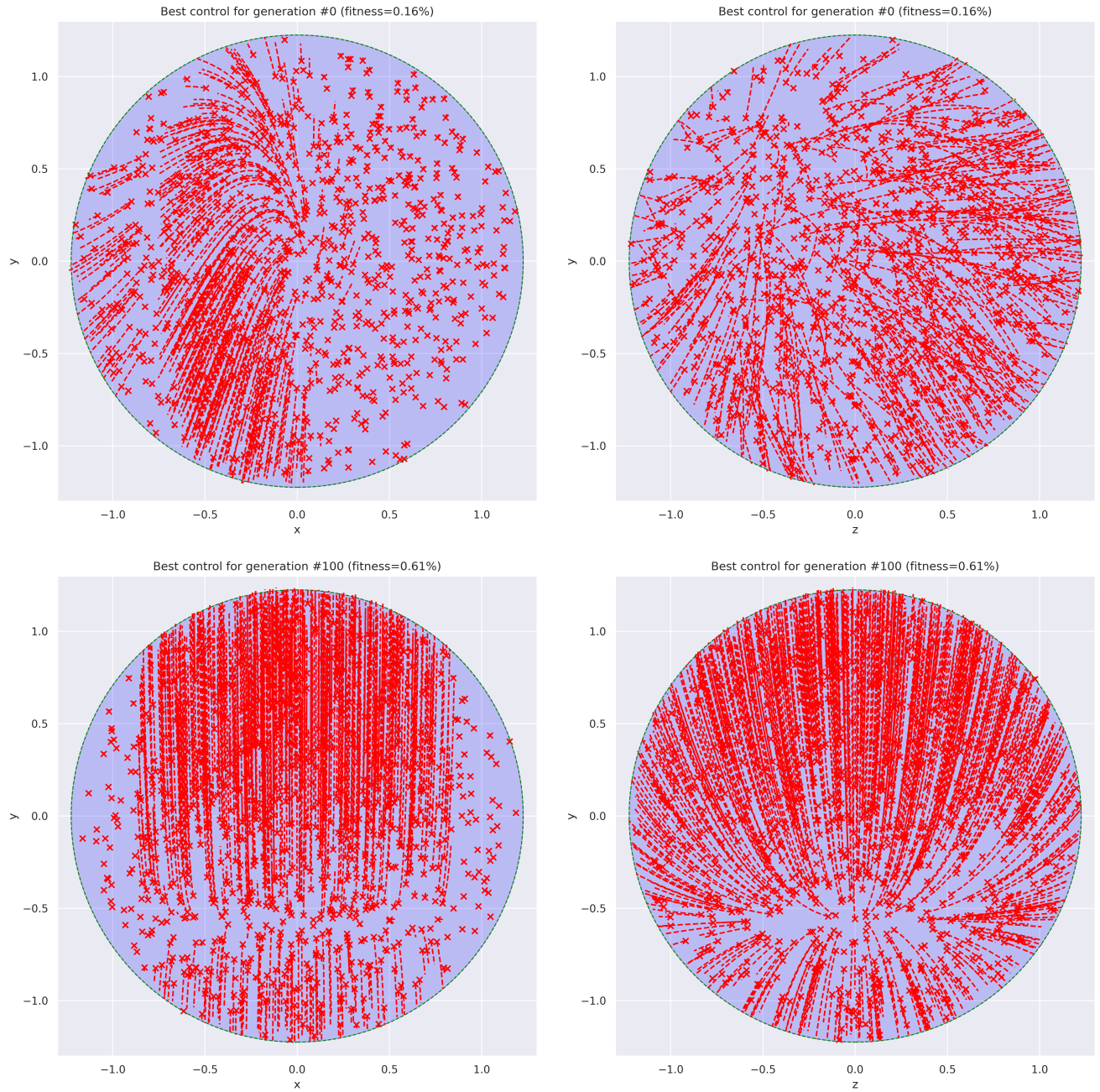

Figure 4: Analysis of individuals for the 3D-Sphere case study, using 1,000 randomly generated initial conditions, unseen during the evolutionary optimization. The two top plots show trajectories for the best control of generation 0, the bottom two show the trajectories for the same initial conditions for the best control at generation 100. Even though the control at generation 100 is able to keep the trajectories viable for longer, they all eventually escape the constraints.

of finding explicit analytic equations for control laws is posed, and a possible framework to solve it is introduced. The proposed approach is based on the evolution of GP trees modeling equations for each control law, while the fitness value of a candidate solution is assessed by sampling the state space of the problem for random initial conditions at each generation. Preliminary experimental results on classical viability benchmarks show that the approach is able to deliver promising results, even for controls including several equations. Having explicit, high-quality control equations

for a viability problem could help improve the communication with the stakeholders and ease the analysis of a problem's behavior.

A possible extension for real-world applications would be to integrate a term into the fitness function to drive the evolutionary search towards control laws that are similar to stakeholders' current control practices. This way small changes, more acceptable with respect to the current practices, could be proposed to them.

Other future works will focus on improving the fitness function. Instead of using randomly generated valid initial conditions at each generation, it might be possible to progressively store information on the viability of initial conditions previously evaluated, and eventually approximate the frontier of the viability kernel during the evolution. As the frontier, which is in principle unknown, is where the initial conditions most interesting for the evaluation of candidate control equations lay, this could provide more relevant information for the evolutionary algorithm, and at the same time deliver interesting results to analyze for viability experts.

A non-negligible part of viability theory is concerned with tyches, in other words, unpredictable and uncontrollable factors that are usually assumed to be stochastic. The same proposed approach described in this work could be used to evolve equations describing deterministic tyches, which would attempt to rapidly push the system towards a non-viable state, thus leading to the discovery of worst-case scenarios for given viability problems. It would also be possible to frame the problem as a competitive co-evolutionary optimization task, with a population of candidate tyches trying to push trajectories to a non-viable state, and a population of control laws attempting to keep them inside the viable area.

Besides a few early attempts [30], more focused on the discovery of an optimal trajectory when starting from an initial condition inside the viability kernel, the application of evolutionary computation to viability theory is still a relatively unexplored field, with several exciting possibilities for evolutionary approaches to shine and provide valuable results for the viability research community.

REFERENCES

[1] Isabelle Alvarez and Sophie Martin. 2011. Geometric robustness of viability kernels and resilience basins. In *Viability and Resilience of Complex Systems: Concepts, Methods and Case Studies from Ecology and Society*. Guillaume Deffuant and Nigel Gilbert, (Eds.) Springer Berlin Heidelberg, Berlin, Heidelberg, 193–218.

[2] Isabelle Alvarez, Laetitia Zaleski, Jean-Pierre Briot, and Marta de A. Irving. 2023. Collective management of environmental commons with multiple usages: a guaranteed viability approach. *Ecological Modelling*, 475, 110186. DOI: https://doi.org/10.1016/j.ecolmodel.2022.110186.

[3] J.-P. Aubin. 1991. *Viability Theory*. Birkhäuser, Basel.

[4] Jean Pierre Aubin. 1997. *Dynamic economic theory: a viability approach*. Vol. 5. Springer Verlag.

[5] Jean-Pierre Aubin, Alexandre Bayen, and Patrick Saint-Pierre. 2011. *Viability Theory: New Directions*. Springer, 830. DOI: 10.1007/978-3-642-16684-6.

[6] Jean-Pierre Aubin and Halina Frankowska. 1985. Heavy viable trajectories of controlled systems. *Annales de l'Institut Henri Poincaré C, Analyse non linéaire*, 2, 5, 371–395. DOI: https://doi.org/10.1016/S0294-1449(16)30397-3.

[7] C. Bernard and S. Martin. 2012. Building strategies to ensure language coexistence in presence of bilingualism. *Applied Mathematics and Computation*, 218, 17, 8825–8841. DOI: https://doi.org/10.1016/j.amc.2012.02.041.

[8] Erick Cantú-Paz. 2004. Adaptive sampling for noisy problems. Tech. rep. Lawrence Livermore National Lab. (LLNL), Livermore, CA (United States).

[9] S. Carpenter, W. Brock, and P. Hanson. 1999. Ecological and social dynamics in simple models of ecosystem management. *Conservation Ecology*, 3(2). http://www.consecol.org/vol3/iss2/art4/.

[10] Duc-Cuong Dang and Per Kristian Lehre. 2015. Efficient Optimisation of Noisy Fitness Functions with Population-based Evolutionary Algorithms. en. In *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*. ACM, Aberystwyth United Kingdom, (Jan. 2015), 62–68. ISBN: 978-1-4503-3434-1. DOI: 10.1145/2725494.2725508.

[11] M. Falcone and P. Saint-Pierre. 1987. Slow and quasi-slow solutions of differential inclusions. *Nonlinear Analysis: Theory, Methods & Applications*, 11, 3, 367–377. DOI: https://doi.org/10.1016/0362-546X(87)90052-6.

[12] J. Michael Fitzpatrick and John J. Grefenstette. 1988. Genetic Algorithms in Noisy Environments. *Machine Learning*, 3, 2/3, 101–120. DOI: 10.1023/A:1022654003254.

[13] Marco Gaudesi, Elio Piccolo, Giovanni Squillero, and Alberto Tonda. 2016. Exploiting evolutionary modeling to prevail in iterated prisoner's dilemma tournaments. *IEEE Transactions on Computational Intelligence and AI in Games*, 8, 3, (Sept. 2016), 288–300. DOI: 10.1109/tciaig.2015.2439061.

[14] David Goldberg E., Kalyanmoy Deb, and James H. Clark. 1992. Genetic algorithms, noise, and the sizing of populations. *Complex Systems*, 6, 333–362.

[15] John J. Grefenstette and J. Michael Fitzpatrick. 1985. Genetic search with approximate function evaluation. In *Proceedings of the 1st International Conference on Genetic Algorithms*. L. Erlbaum Associates Inc., USA, 112–120. ISBN: 0805804269.

[16] E. Hairer, S. P. Norsett, and G. Wanner. 1993. *Solving Ordinary Differential Equations I: Nonstiff problems*. Springer Berlin Heidelberg. DOI: 10.1007/978-3-540-78862-1.

[17] Baligh Al-Helali, Qi Chen, Bing Xue, and Mengjie Zhang. 2020. Genetic Programming with Noise Sensitivity for Imputation Predictor Selection in Symbolic Regression with Incomplete Data. In *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, Glasgow, United Kingdom, (July 2020), 1–8. ISBN: 978-1-72816-929-3. DOI: 10.1109/CEC48606.2020.9185526.

[18] Torsten Hildebrandt and Jürgen Branke. 2015. On Using Surrogates with Genetic Programming. *Evolutionary Computation*, 23, 3, (Sept. 2015), 343–367. DOI: 10.1162/EVCO_a_00133.

[19] Gregory Hornby, Al Globus, Derek Linden, and Jason Lohn. 2006. Automated antenna design with evolutionary algorithms. In *Space 2006*. American Institute of Aeronautics and Astronautics, (Sept. 2006). DOI: 10.2514/6.2006-7242.

[20] Yaochu Jin and J. Branke. 2005. Evolutionary optimization in uncertain environments - a survey. *IEEE Transactions on Evolutionary Computation*, 9, 3, 303–317. DOI: 10.1109/TEVC.2005.846356.

[21] John R Koza. 1994. Genetic programming as a means for programming computers by natural selection. *Statistics and computing*, 4, 87–112.

[22] Fiacc Larkin and Conor Ryan. 2009. Avoiding the pitfalls of noisy fitness functions with genetic algorithms. en. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. ACM, Montreal Québec Canada, (July 2009), 1861–1862. ISBN: 978-1-60558-325-9. DOI: 10.1145/1569901.1570204.

[23] Benoit Leblanc and Evelyne Lutton. 1998. Bitwise regularity and ga-hardness. In *ICEC 98, May 5-9, Anchorage, Alaska*.

[24] E. Lutton and J. Lévy Véhel. 1998. Hölder functions and deception of genetic algorithms. *IEEE transactions on Evolutionary computation*, 2, 2, (July 1998), 56–72.

[25] E. Lutton and J. Lévy Véhel. 2006. Pointwise regularity of fitness landscapes and the performance of a simple es. In *CEC'06*. Vancouver, Canada, (July 2006).

[26] Jordan MacLachlan, Yi Mei, Juergen Branke, and Mengjie Zhang. 2020. Genetic Programming Hyper-Heuristics with Vehicle Collaboration for Uncertain Capacitated Arc Routing Problems. *Evolutionary Computation*, 28, 4, (Dec. 2020), 563–593. DOI: 10.1162/evco_a_00267.

[27] S. Martin. 2004. The cost of restoration as a way of defining resilience: a viability approach applied to a model of lake eutrophication. *Ecology and Society*, 9, 2, 19. https://hal.inrae.fr/hal-02583472.

[28] S. Martin and I. Alvarez. 2019. Anticipating shocks in the state space: characterizing robustness and building increasingly robust evolutions. *SIAM Journal on Control and Optimization*, 57, 1, 490–509. eprint: https://doi.org/10.1137/16M1061175. DOI: 10.1137/16M1061175.

[29] JD. Mathias, J. Anderies, and Janssen. 2017. On our rapidly shrinking capacity to comply with the planetary boundaries on climate change. *Scientific Report*, 7, 42061. DOI: https://doi.org/10.1038/srep42061.

[30] Salma Mesmoudi, Nathalie Perrot, Romain Reuillon, Paul Bourgine, and Evelyne Lutton. 2010. Optimal viable path search for a cheese ripening process using a multi-objective ea. In *ICEC 2010, International Conference on Evolutionary Computation*. 24-26 oct, Valencia, Spain. (Oct. 2010).

[31] Aaron Meurer et al. 2017. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3, (Jan. 2017), e103. DOI: 10.7717/peerj-cs.103.

[32] Brad L. Miller and David E. Goldberg. 1995. Genetic algorithms, tournament selection, and the effects of noise. *Complex Syst.*, 9.

[33] Luis F. Miranda, Luiz Otavio V. B. Oliveira, Joao Francisco B. S. Martins, and Gisele L. Pappa. 2017. How noisy data affects geometric semantic genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference* (GECCO '17). Association for Computing Machinery, Berlin, Germany, 985–992. ISBN: 9781450349208. DOI: 10.1145/3071178.3071300.

[34] Aïchouche Oubraham and Georges Zaccour. 2018. A survey of applications of viability theory to the sustainable exploitation of renewable resources. *Ecological Economics*, 145, 346–367. DOI: https://doi.org/10.1016/j.ecolecon.2017.11.008.

[35] Alain Ratle. 1998. Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature* (PPSN V). Springer-Verlag, Berlin, Heidelberg, 87–96. ISBN: 3540650784.

[36] Magnus Rattray and Jonathan Shapiro. 1996. Noisy fitness evaluation in genetic algorithms and the dynamics of learning. In *Proceedings of the 4th Workshop on Foundations of Genetic Algorithms. San Diego, CA, USA, August 5 1996*. Richard K. Belew and Michael D. Vose, (Eds.) Morgan Kaufmann, 117–139.

[37] Patrick Saint-Pierre. 1994. Approximation of the viability kernel. *Applied Mathematics and Optimization*, 29, 2, 187–209.

[38] Yasuhito Sano and Hajime Kita. 2002. Optimization of Noisy Fitness Functions by means of Genetic Algorithms using History of Search. en. *IEEJ Transactions on Electronics, Information and Systems*, 122, 6, 1001–1008. DOI: 10.1541/ieejeiss 1987.122.6_1001.

[39] Michael Schmidt and Hod Lipson. 2009. Distilling free-form natural laws from experimental data. *science*, 324, 5923, 81–85.

[40] Sara Silva, Leonardo Vanneschi, Ana I.R. Cabral, and Maria J. Vasconcelos. 2018. A semi-supervised genetic programming method for dealing with noisy labels and hidden overfitting. *Swarm and Evolutionary Computation*, 39, 323–338. DOI: https://doi.org/10.1016/j.swevo.2017.11.003.

[41] Fatima-Zahra Tani, Alain Rapaport, and Térence Bayen. 2019. A hybrid control against species invasion in the chemostat. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, 2814–2819. DOI: 10.1109/CDC40024.2019.9029692.

[42] Pauli Virtanen et al. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17, 261–272. DOI: 10.1038/s41592-019-0686-2.