Interpretable AI for policy-making in pandemics

Leonardo Lucio Custode
leonardo.custode@unitn.it
University of Trento
Trento, Italy

Giovanni Iacca
giovanni.iacca@unitn.it
University of Trento
Trento, Italy

ABSTRACT

Since the first wave of the COVID-19 pandemic, governments have applied restrictions in order to slow down its spreading. However, creating such policies is hard, especially because the government needs to trade-off the spreading of the pandemic with the economic losses. For this reason, several works have applied machine learning techniques, often with the help of special-purpose simulators, to generate policies that were more effective than the ones obtained by governments. While the performance of such approaches are promising, they suffer from a fundamental issue: since such approaches are based on black-box machine learning, their real-world applicability is limited, because these policies cannot be analyzed, nor tested, and thus they are not trustable. In this work, we employ a recently developed hybrid approach, which combines reinforcement learning with evolutionary computation, for the generation of interpretable policies for containing the pandemic. These policies, trained on an existing simulator, aim to reduce the spreading of the pandemic while minimizing the economic losses. Our results show that our approach is able to find solutions that are extremely simple, yet very powerful. In fact, our approach has significantly better performance (in simulated scenarios) than both previous work and government policies.

KEYWORDS

COVID-19, interpretable AI, reinforcement learning

ACM Reference Format:
Leonardo Lucio Custode and Giovanni Iacca. 2022. Interpretable AI for policy-making in pandemics. In *Genetic and Evolutionary Computation Conference Companion (GECCO '22 Companion), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3520304.3533959

1 INTRODUCTION

COVID-19 has changed the way of living of all the people on Earth. In fact, since its outbreak, several countries adopted safety measures such as: social distancing, lockdowns, closing certain types of economic activities and others. These safety measures were taken as a prevention mechanism to slow down the spreading of the pandemic in the world population. However, some safety measures may have a relevant impact on the economy of a country. For this reason, it is important to find a trade-off between spreading of the pandemic and economic losses.

To this end, some previous works focused on the use of simulators to estimate the virus propagation and economic losses given a policy [8, 17]. In this setting, a policy decides the safety measures to adopt in a given scenario. However, while these works have been proven to be able to find policies that have better trade-offs between economic losses and pandemic spread than the ones used by governments, they do not employ interpretable AI methods. Thus, even though such models do perform very well, their applicability is limited due to the lack of understandability. More specifically, the main drawbacks of non-interpretable approaches for this task are: a) the fact that the trained models act as "oracles" and, thus, a policymaker cannot understand the rationale underlying a decision; and b) the fact that such models cannot be inspected and, thus, their behavior cannot be formally specified.

In this work, we propose interpretable models for policy-making in pandemics. These models, which combine decision trees (DTs) trained by means of grammatical evolution with Q-learning, are very simple and objectively interpretable. Moreover, our solutions exhibit better performance w.r.t. non-interpretable state-of-the-art models. Since our evolved policies are both more effective and more interpretable than existing black-box models, they are extremely suitable for creating policies to control the pandemic while avoiding unnecessary economic damage.

This paper is structured as follows. The next section makes a short review of the state of the art. Section 3 describes the method used to evolve interpretable DTs. In Section 4 we show the experimental results and, in Section 5, we interpret the trees obtained. Finally, in Section 6 we draw the conclusions of this work and suggest future work.

2 RELATED WORK

2.1 Policy-making for pandemics

In [8], the authors propose a simulator of pandemics that is able to simulate a population at a very fine granularity, modeling the activities that each agent can perform, as well as various types of economic activities. The goal of this simulator is to test policies with the objective of minimizing simultaneously the spreading of the pandemic and the economic damages. The authors test various handcrafted policies, policies used from a few countries, and a deep reinforcement learning based policy. The results show that the deep reinforcement learning based policy is able to outperform both the handmade policies and the governmental ones.

In [17], the authors propose a simulator to estimate the effects on the management of closing economic activities on both the spreading of the pandemic and the economic losses. The proposed

GECCO '22 Companion, July 9–13, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9268-6/22/07…$15.00
https://doi.org/10.1145/3520304.3533959

simulator is tailored on the U.S. economy, simulating all the 51 states and a central entity that manages subsidies.

The main difference between the approaches proposed above is that in [8] the simulator acts at a very low scale, but with a high level of detail, being able to simulate aspects of everyday life, while in [17] the simulator aims to perform at a very large scale with a lower level of detail.

In [10], the authors use a surrogate-assisted evolutionary process to optimize an agent that has to apply restrictions to avoid the spreading of the pandemic. They make use of two neural networks: one that acts as the policy, and the other that estimates the quality of a policy, trained on real data. While the performance of this approach is promising, the use of black-box policies such as neural network makes the real-world application of such systems difficult [12].

In cite [1], the authors propose an agent-based model to simulate the outcome of restrictions on the spread of the pandemic. The proposed model can be accelerated on GPUs for faster computation. Moreover, the authors test a number of policies on the agent-based model to discover effective ways to contrast the pandemic. While the proposed agent-based model is quite comprehensive, it lacks the capability of simulating school and work closures, which may be essential to obtain effective policies.

2.2 Interpretable AI

In recent years, the need for interpretable AI emerged. Interpretable AI [2] is defined as the branch of AI that focuses on models that are inherently interpretable by humans. This type of models is extremely important in high-stakes and safety critical scenarios, as the ability to inspect the model and understand its behavior becomes crucial [12].

While the field of interpretable AI is making progresses, there are some sub-fields that are still seen as crucial to the development of interpretable AI. One of these fields is interpretable reinforcement learning [13].

Lately, several works approached the problem of interpretable reinforcement learning.

Silva et al. [16] make use of differentiable DTs that are trained by means of the proximal policy optimization algorithm [15]. While this approach has shown very promising performance in the tested environments, its performance degrades significantly when transforming the differentiable DT into a regular DT.

Dhebar et al. [6] propose an evolutionary approach for producing DTs with nonlinear splits (i.e., hyperplanes defined by conditions) for reinforcement learning tasks. This approach is able to obtain very good performance in the tested tasks. However, the high non-linearity of the splits hinders the interpretability of the DTs obtained.

In [3], the authors propose a two-level optimization approach that combines grammatical evolution [14] and Q-learning [19] for evolving DTs for reinforcement learning tasks. The approach is tested on classic reinforcement learning tasks, and the results show that the systems obtained are competitive w.r.t. non-interpretable state-of-the-art models while being very easy to interpret. However, this approach only works in environments with discrete action spaces.

In [4], the authors extend the approach performed in [3] to make it work in scenarios with continuous action spaces. To do so, they make use of a co-evolutionary approach [11] that allows to evolve simultaneously decision trees and "pools" of continuous actions.

In [5], the authors use genetic programming [9] and CMA-ES [7] to evolve interpretable DTs that are able to work in RL settings with images as input. However, the experimental results show that the proposed approach exhibits good performance only in scenarios that are not affected by noise.

3 METHOD

3.1 Simulator

Since our goal is to evolve general policies for minimizing the spread of the pandemic and, at the same time, reducing the economic losses, we employ the simulator proposed in [8]. In fact, this simulator allows us to test rules that are applicable in every country (i.e., not only tailored to U.S. as in [17]) and that do not make use of economic subsidies.

Below, we briefly summarize the state, actions and reward used in the simulator. For further details on the simulation, we refer the interested reader to [8].

3.1.1 State. The simulator, at each step, provides the following features:

- i_g: Number of infected people since the beginning of the simulation;
- r_g: Number of recovered people since the beginning of the simulation;
- c_g: Number of patients in critical conditions since the beginning of the simulation;
- d_g: Number of deaths since the beginning of the simulation;
- n_g: Number of people that did not contract the virus from the beginning of the simulation until the current simulation day;
- i_d: Number of daily infected;
- r_d: Number of daily recovered;
- c_d: Number of patients that are in critical conditions in the current simulation day;
- d_d: Number of deaths occurring in the current simulation day;
- n_d: Number of people that did not contract the virus in the current day[1];
- l: The current level of ongoing restrictions;
- h: A Boolean variable indicating whether the capacity of the hospitals is saturated.

All the variables are normalized by using a min-max normalization before being fed to the policy-making agent. Note that the simulator returns a noisy version of the estimates of the variables described above, to simulate a real-world scenario in which not all the results of the tests are known.

3.1.2 Actions. The agent can choose an action between a pool of 5 actions:

(1) Stage 0: No restrictions
(2) Stage 1: Stay at home if sick, gathering limits:

[1]Please note that the simulator always returns $n_d = n_g$. However, for consistency with the experiments reported in [8], in our experiments we kept both values as inputs to the policy-making agent.

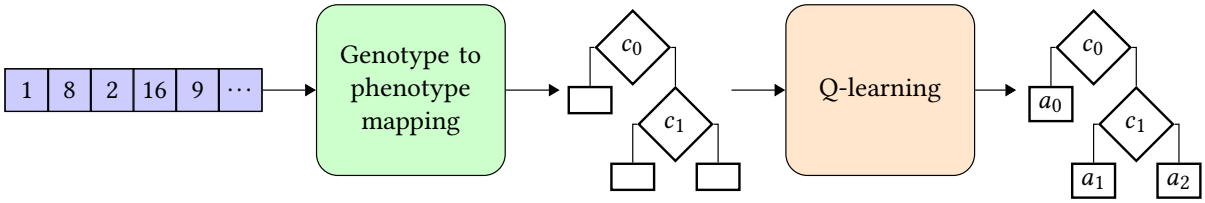

Figure 1: Graphical representation of the fitness evaluation phase.

- Low-risk people: 50
- High-risk people: 25

(3) Stage 2: Limitations of stage 1 + wear masks + light social distancing, schools and hair salons are closed, gathering limits:
- Low-risk people: 25
- High-risk people: 10

(4) Stage 3: Limitations of stage 2 + moderate social distancing + no gatherings

(5) Stage 4: Limitations of stage 3 + heavy social distancing + office and retail stores closed

A more detailed list of actions is described in [8].

3.1.3 Reward. At each step, the agent receives a reward of:

$$r = -0.4 \cdot max(\frac{c_d - C}{C}, 0) - 0.1 \cdot \frac{l^{1.5}}{5^{1.5}}, \quad (1)$$

where C is the capacity of the hospitals. Note that this negative reward is to be maximized.

Essentially, this reward function aims to trade-off the number of patients in critical conditions with the stringency level of the restrictions. Moreover, this function indirectly induces the agent to minimize also the closing of stores, limiting the economic losses.

3.2 Evolution of decision trees

Since the action space of this environment is discrete and the interpretability of the policies is crucial for their application in real world scenarios, we employ the method described in [3] for evolving policies under the form of DTs.

Thus, we use grammatical evolution to evolve DTs with empty leaves, i.e., leaves that do not perform any action. In the fitness evaluation phase, the leaves are then trained by means of Q-learning [19], by exploiting the reward signals coming from the environment.

A graphical illustration of the fitness evaluation phase is shown in Figure 1.

The evolutionary process is iterated for 50 generations using a population of 45 individuals. All the parameters (including the ones shown in the following subsections, e.g., mutation rate and tournament size) were empirically tuned to balance performance and computational cost. We perform 10 i.i.d. experimental runs of the evolutionary process to assess the statistical repeatability of our results.

3.2.1 Individual encoding. Each individual is encoded as a list of integers, ranging from 0 to a maximum value M, such that M is significantly higher than the maximum numbers of productions in the grammar. In our experiments, $M = 4 \cdot 10^3$.

3.2.2 Genotype to phenotype mapping. Each genotype (i.e., a list of integers) is transformed into its corresponding phenotype, i.e., a Python program that encodes the corresponding DT by means of "if-then-else" statements. To do so, we make use of the grammar shown in Table 1. Starting from the first gene and from a production containing only the rule "dt", each gene is used to replace the first rule currently found in the string with the $(i \ mod \ k)$-th production, where i is the value of the current gene and k is the size of the production of the current rule (i.e., the number of choices associated to the current rule).

Note that the leaves of the resulting tree are not defined, i.e., they do not have a fixed action. This allows us to perform Q-learning in the fitness evaluation phase, as mentioned above.

3.2.3 Selection. To select the individuals for mutation/crossover, we use a tournament selection with tournament size 2.

3.2.4 Mutation. The mutation operator used in our experiments is the uniform mutation, which replaces the current gene with a new integer uniformly sampled in $[0, M]$. The mutation operator is applied with probability 1 (i.e., to all the individuals in the population), while the mutation rate (e.g., the probability of mutation of each gene) is 10%.

3.2.5 Crossover. Since crossover proved to be too "destructive" in preliminary experiments, we do not employ it in the experiments reported below. By destructive, we mean that, by means of this operator, the offspring can be significantly different from both parents, hindering the exploitation of good structures emerged during the evolutionary process.

3.2.6 Replacement. In order to increase the exploitativeness of our approach, we introduce a replacement operator. This operator replaces a parent with an offspring only in case the offspring has (strictly) better fitness than the parent.

3.2.7 Fitness evaluation. The fitness of each phenotype is determined by using the corresponding policy to perform decisions in the `PandemicSimulator`[2] environment. To learn actions for the leaves, we employ ε-greedy Q-learning, with learning rate $\alpha = 10^{-3}$, $\varepsilon = 0.05$, and random initialization for the Q-values in $[-1, 1]$. Moreover, we evaluate each phenotype in 10 independent episodes, each of which simulates 100 days of pandemic.

4 RESULTS

Table 2 shows the scores obtained by the best agents obtained in each independent run. The "Seed" column indicates the seed used in

[2]https://github.com/SonyAI/PandemicSimulator

Table 1: Grammar used to produce the decision trees.

Rule	Production
dt	$\langle if \rangle$
if	$if\ \langle condition \rangle\ then\ \langle action \rangle\ else\ \langle action \rangle$
condition	$input\_var\ \langle comp_op \rangle\ \langle const_{input\_var} \rangle$
action	$leaf \mid \langle if \rangle$
comp_op	$lt \mid gt$
$const_{1..10}$	[0, 1) with step 0.1
$const_{11,12}$	[0, 1) with step 0.5

each of the 10 i.i.d. runs. This value is used to seed the evolutionary process. The simulator, instead, is seeded at each episode by using the index of the episode as the seed. Here, we differentiate between *training* and *testing* returns. Training returns are defined as the returns (i.e., the sum of the rewards obtained at each timestep of an episode) obtained by the agent in the training process, while testing refers to the returns obtained by the agent after the evolutionary process, i.e., when learning is disabled. Moreover, we also measure the *interpretability* of our solutions. To do so, we employ the metric of interpretability proposed in [18], reworked as done in [3]. This metric is defined as:

$$\mathcal{M} = -0.2 + 0.2\ell + 0.5n_o + 3.4n_{nao} + 4.5n_{naoc}$$

where ℓ is the number of symbols appearing in the formula, n_o is the number of operations contained in the model, n_{nao} is the number of non-arithmetical operations, and n_{naoc} is the maximum number of consecutive compositions of non-arithmetical operations. Essentially, this metric uses as proxy for interpretability the *complexity* of the model. Lower values of $\mathcal{M}$ correspond to simpler models, which in principle may be more interpretable (a constant model would have $\mathcal{M} = 0$). Note that, in general, the optimal value of $\mathcal{M}$ cannot be determined a priori for a given task. Note that, since a split (i.e., a condition of the tree) is a non-arithmetical operation, it increases both n_{nao} and (potentially) n_{naoc}, thus adding a single split to the tree may significantly increase the $\mathcal{M}$ score. For instance, while the decision trees obtained from seeds 1 and 8 contain only one split and they achieve a $\mathcal{M} = 17.8$ (with the same test return), adding another split (e.g., seeds 2 and 3) results in doubling the value of $\mathcal{M}$. It is important to note that we use the $\mathcal{M}$ metric to assess, a-posteriori, the interpretability of the solutions that were discovered by the evolutionary process. Moreover, no bloat-control mechanism has been implemented to minimize $\mathcal{M}$.

Figure 2 compares the results obtained with the best DT produced by our method (i.e., the one corresponding to seed 3 in Table 2, which obtained the highest test return) to the policies reported in [8]. In particular, the policies used for the comparison are the following:

(1) S0-4-0: Starts with stage 0, then, once 10 have been infected, it switches to stage 4 and, after 30 days, it returns to stage 0.
(2) S0-4-0FI: Similar to S0-4-0, but the return from stage 4 to stage 0 is performed gradually, by reducing the restrictions by 1 stage each 5 days.
(3) S0-4-0GI: Similar to S0-4-0, but the intermediate stages from stage 4 to stage 0 last 10 days instead of 5.
(4) S0: Always applies stage 0 restrictions.
(5) S1: Always applies stage 1 restrictions.
(6) S2: Always applies stage 2 restrictions.
(7) S3: Always applies stage 3 restrictions.
(8) S4: Always applies stage 4 restrictions.

We test all the policies in 30 independent episodes. From Figure 2, we can see that our best DT outperforms all the hand-crafted policies under comparison. The statistical difference between the best decision tree evolved and the hand-crafted policies has been confirmed by a two-sided Wilcoxon test with $\alpha = 0.05$.

As for the deep reinforcement learning based policy presented in [8], while we could not test it due to the fact that the model is not publicly available (and neither its numerical results are), we estimate, from the plots reported in the original paper, an average return of about -5 (See Figure 6.p (Learned) in [8]), which is substantially lower than the return obtained by our best DT.

Finally, in Figure 3 we compare the policy obtained by means of our best DT with the ones implemented by the Italian and Swedish governments (for which we use the implementation provided in [8]). These policies act as follows:

(1) ITA (approximation of the restrictions adopted by the Italian government): Increase the restrictions gradually from stage 0 to stage 4 and then it gradually returns to stage 2.
(2) SWE (approximation of restrictions adopted by the Swedish government): Applies stage 0 regulations for 3 days and then stage 1 restrictions for all the duration of the simulation.

Also in this case, we observe that the performance are significantly better than the compared policies. Similarly, also in this case, a Wilcoxon test with $\alpha = 0.05$ confirms the statistical significance of the results.

Moreover, we observe that the number of people infected, by using our proposed policy, is significantly smaller w.r.t. the other approaches. This fact, combined with the very high returns obtained by our policy, suggests that the policy minimizes the economic losses by trying to stop the pandemic at the beginning, and then making the restrictions less stringent.

In the next section, we analyze the policy to understand the reasons underlying its significantly higher performance.

Table 2: Best results obtained in 10 i.i.d. experimental runs of the proposed evolutionary process. Train and test returns are averaged over 10 independent episodes. Underlined results indicate the best value in each column.

Seed	Train return	Test return	$\mathcal{M}$
0	<u>-1.39</u>	-2.43	53.40
1	-2.07	-1.45	<u>17.80</u>
2	-1.48	-1.47	35.60
3	-2.04	<u>-1.11</u>	35.60
4	-2.07	-2.31	53.40
5	-1.99	-1.45	35.60
6	-2.06	-2.24	53.40
7	-1.97	-1.46	53.40
8	-1.63	-1.45	<u>17.80</u>
9	-1.48	-1.36	53.40

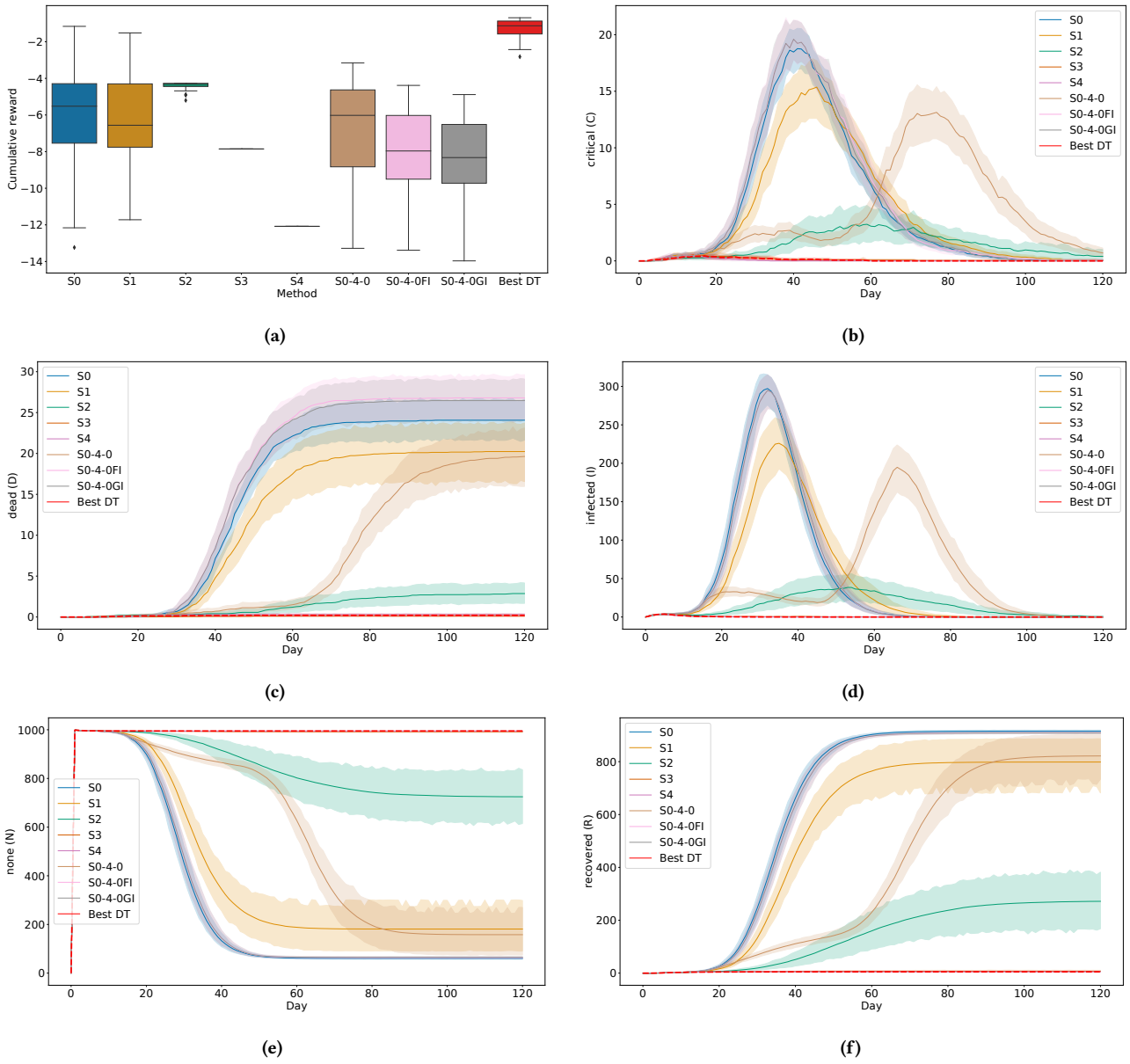

Figure 2: Comparison of the performance of the best DT evolved w.r.t. handcrafted policies proposed in [8]: (a) returns (cumulative rewards at each step of the simulation) obtained by the policies; (b) number of patients in critical conditions for each simulated day; (c) number of deaths since the beginning of the simulation; (d) number of infected people for each simulated day; (e) number of never-infected people since the beginning of the simulation; (f) number of recovered patients since the beginning of the simulation.

5 INTERPRETATION

The best DT obtained is shown in Figure 4. While the size of the DT is quite limited, its performance are extremely satisfactory. We can see that the DT makes use of two conditions to compute its output. The decision tree has been simplified by removing branches that are never taken and merging (sibling) leaves that contain the same action.

The first condition (the root) checks whether the number of never-infected people (n_d) is greater than the 90%. If so, then it checks the second condition. Otherwise, it applies stage 2 restrictions, regardless of the other variables. We may hypothesize that, in the latter case, the policy does that because, in case the number of infected people (from the beginning of the pandemic) is greater than the 10%, then the virus may spread very quickly if the restrictions are not appropriate.

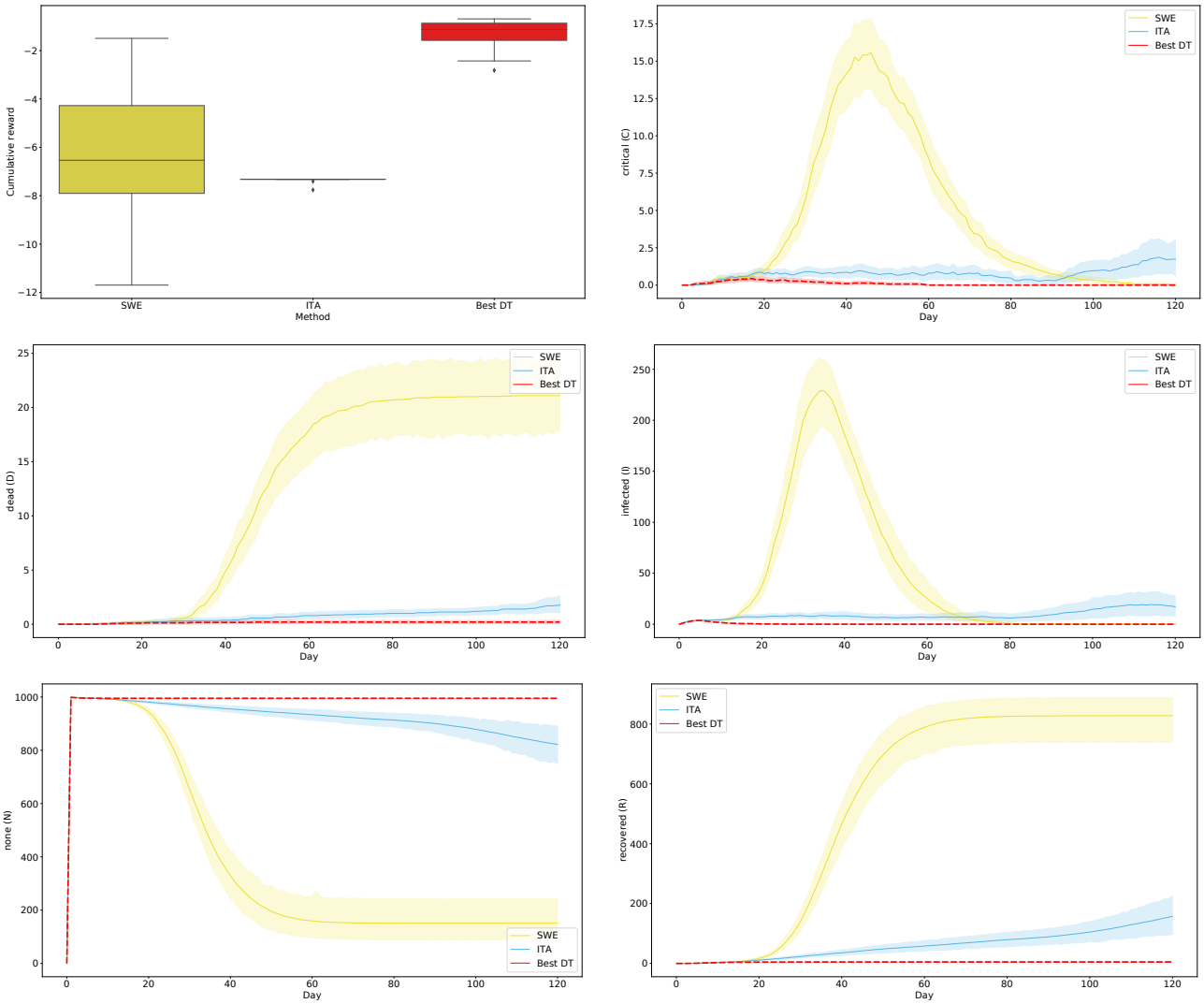

Figure 3: Comparison of the performance of the best DT evolved w.r.t. policies adopted by the Swedish (SWE) and Italian (ITA) governments, as implemented in [8]: (a) returns (cumulative rewards at each step of the simulation) obtained by the policies; (b) number of patients in critical conditions for each simulated day; (c) number of deaths since the beginning of the simulation; (d) number of infected people for each simulated day; (e) number of never-infected people since the beginning of the simulation; (f) number of recovered patients since the beginning of the simulation.

The second condition, instead, checks the number of people infected so far (i_g). If the known number of infected is greater than zero, it applies stage 3 restrictions. Otherwise, it applies no restrictions.

In essence, the combination of the two branches of the root aims to induce a "strong" response to the initial wave of the pandemic, which is then slightly relaxed after the number of total cases increases. Thus, the overall strategy learned by the agent is to stop the pandemic at the beginning, keeping slightly less stringent restriction after the initial wave, to avoid new waves.

While the fact that a simple policy like the one shown in Figure 4 outperforms a deep neural network is counter-intuitive, we hypothesize that it may be the case that in some tasks the extremely high complexity of deep neural networks, combined with the fact that gradient-based methods are more prone to get stuck in local optima, makes their training harder than training very simple models.

6 CONCLUSIONS

In this paper, we leverage interpretable AI methodologies to train interpretable policies for managing a pandemic.

Despite the widely-thought trade-off between interpretability and performance, the obtained policies proved to be significantly

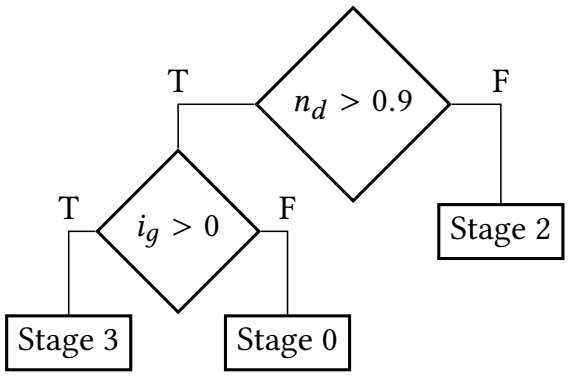

Figure 4: Best decision tree (on the test score) evolved.

better than handcrafted policies, deep learning policies, and government policies.

It is important to note that the results shown above have been tested in a simulated scenario and, thus, their applicability to real-world scenario must be assessed, involving in such evaluation experts in pandemic control.

A limitation of the current work is the simulation scenario adopted: in our work, we simulate a population of 1000 people. The assessment of the scalability of our results is left as future work. Likewise, it would be important to extend the simulations to account for, among other things, different economies, different types of virus (e.g., with different propagation speeds and mortality rates), as well as various levels of population density and other demographic aspects, such as age and gender distribution.

Other future works include: testing the proposed methodology on different simulators with different objectives, and adopting more realistic scenarios w.r.t. economical aspects (e.g., subsidies) or epidemiological aspects (e.g., the possibility of the emergence of new variants).

REFERENCES

[1] Khalil Al Handawi and Michael Kokkolaras. 2021. Optimization of Infectious Disease Prevention and Control Policies Using Artificial Life. *IEEE Transactions on Emerging Topics in Computational Intelligence* 6 (2021), 26–40. Issue 1.

[2] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion* 58 (June 2020), 82–115. https://doi.org/10.1016/j.inffus.2019.12.012

[3] Leonardo Lucio Custode and Giovanni Iacca. 2020. Evolutionary learning of interpretable decision trees.

[4] Leonardo Lucio Custode and Giovanni Iacca. 2021. A co-evolutionary approach to interpretable reinforcement learning in environments with continuous action spaces. In *Symposium Series on Computational Intelligence (SSCI)*. IEEE, New York, NY, USA, 1–8.

[5] Leonardo Lucio Custode and Giovanni Iacca. 2022. Interpretable pipelines with evolutionarily optimized modules for RL tasks with visual inputs.

[6] Yashesh Dhebar, Kalyanmoy Deb, Subramanya Nageshrao, Ling Zhu, and Dimitar Filev. 2020. Interpretable-AI Policies using Evolutionary Nonlinear Decision Trees for Discrete Action Systems. http://arxiv.org/abs/2009.09521

[7] Nikolaus Hansen and Andreas Ostermeier. 1996. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *IEEE International Conference on Evolutionary Computation*. IEEE, New York, NY, USA, 312–317.

[8] Varun Kompella*, Roberto Capobianco*, Stacy Jong, Jonathan Browne, Spencer Fox, Lauren Meyers, Peter Wurman, and Peter Stone. 2020. Reinforcement Learning for Optimization of COVID-19 Mitigation policies. arXiv:2010.10560 [cs.LG]

[9] John R. Koza. 1992. *Genetic programming: on the programming of computers by means of natural selection*. MIT Press, Cambridge, Mass.

[10] Risto Miikkulainen, Olivier Francon, Elliot Meyerson, Xin Qiu, Darren Sargent, Elisa Canzani, and Babak Hodjat. 2021. From prediction to prescription: evolutionary optimization of nonpharmaceutical interventions in the COVID-19 pandemic. *IEEE Transactions on Evolutionary Computation* 25, 2 (2021), 386–401.

[11] Mitchell A. Potter and Kenneth A. De Jong. 1994. A cooperative coevolutionary approach to function optimization. In *Parallel Problem Solving from Nature — PPSN III*. Springer, Berlin, Heidelberg, 249–257. https://doi.org/10.1007/3-540-58484-6_269

[12] Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence* 1, 5 (May 2019), 206–215. https://doi.org/10.1038/s42256-019-0048-x

[13] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. 2021. Interpretable Machine Learning: Fundamental Principles and 10 Grand Challenges.

[14] Conor Ryan, Jj Collins, and Michael O Neill. 1998. Grammatical evolution: Evolving programs for an arbitrary language. In *European Conference on Genetic Programming*. Springer, Berlin, Heidelberg, 83–96. https://doi.org/10.1007/BFb0055930

[15] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. http://arxiv.org/abs/1707.06347 arXiv:1707.06347.

[16] Andrew Silva, Matthew Gombolay, Taylor Killian, Ivan Jimenez, and Sung-Hyun Son. 2020. Optimization Methods for Interpretable Differentiable Decision Trees Applied to Reinforcement Learning. In *International Conference on Artificial Intelligence and Statistics*. PMLR, Palermo, Italy, 1855–1865. http://proceedings.mlr.press/v108/silva20a.html

[17] Alexander Trott, Sunil Srinivasa, Douwe van der Wal, Sebastien Haneuse, and Stephan Zheng. 2021. Building a foundation for data-driven, interpretable, and robust policy design using the ai economist.

[18] Marco Virgolin, Andrea De Lorenzo, Eric Medvet, and Francesca Randone. 2020. Learning a Formula of Interpretability to Learn Interpretable Formulas. In *Parallel Problem Solving from Nature – PPSN XVI*, Thomas Bäck, Mike Preuss, André Deutz, Hao Wang, Carola Doerr, Michael Emmerich, and Heike Trautmann (Eds.). Springer International Publishing, Cham, 79–93.

[19] Christopher John Cornish Hellaby Watkins. 1989. *Learning from delayed rewards*. Ph.D. Dissertation. King's College, Cambridge, United Kingdom.