Barbara Luppi[1] – Francesco Parisi[2]

The Economics of Evolutionary Programming

Abstract: In this paper, we present an economic model of evolutionary computation technologies, highlighting how alternative propertization and liability mechanisms shape the balance between "exploration" and "exploitation" in evolutionary research. We consider two interrelated effects of evolutionary experimentation: (i) the positive externalities that occur when superior mutations enhance performance, benefitting all subsequent implementations of the algorithm; (ii) the negative externalities that arise when inferior mutations cause harm. Our analysis reveals that experimentation reaches the optimal stopping point only with a combined use of propertization and liability instruments. If intellectual property frameworks fail to propertize benefits from evolutionary processes, limiting liability may be a second-best means of encouraging experimentation.

Keywords: AI, evolutionary computation, genetic programming, liability, patents
JEL Codes: K32, K41, O31, D62

1. Introduction

Evolutionary computation—automating the process of refining an algorithm to make the algorithm more effective—has greatly expanded the horizons of artificial intelligence, rapidly advancing within the fields of machine learning and optimization. By evolving iteratively—much like in biological evolution—evolutionary computation is reducing complex optimization challenges to discover highly effective solutions in diverse fields such as engineering, economics, and data science. Experimentation in evolutionary algorithms can yield better solutions, but such experimentation often entails navigating suboptimal mutations, which may lead to unintended consequences or undesirable outcomes.

This paper focuses on two representative examples of evolutionary computation that mimic natural selection to find optimal solutions: evolutionary programming and genetic

[1] Associate Professor of Economics at the University of Modena and Reggio Emilia, Department of Economics (Email: barbara.luppi@unimore.it).
[2] Oppenheimer Wolff and Donnelly Professor of Law at the University of Minnesota, and Professor of Economics at the University of Bologna (Email: parisi@umn.edu). We would like to thank Samuel Makikalli for providing research assistance.

programming.[3] As detailed below, both processes involve generating variations of the original program at each stage, selecting only the highest-fitness variations to advance. The evolutionary selection process typically halts upon reaching a predefined fitness level. However, in many applications, the dynamic nature of the experimentation process makes the fitness level a moving target rather than a fixed endpoint, often prompting the need for continuous adaptation.

This dynamism can occur for several reasons. First, perfection is an unattainable goal—there is always room for further improvement, especially as the problem space evolves. Second, the application of the algorithm in specific contexts may introduce new challenges or uncover additional degrees of freedom, potentially shifting the previously established fitness frontier. Additionally, external factors such as changing industry standards, regulatory requirements, or advancements in competitors' technology can raise the bar for what is considered "optimal."

In all these scenarios, an algorithm that is adopted and patented (hereinafter referred to as the "state-of-the-art" algorithm) may be subjected to further experimentation, leading to updates and refinements.[4] Superior mutations, once adopted, establish new benchmarks for future applications and refinements. Determining the ideal stopping point in an evolutionary computation process entails balancing a dual set of effects. While mutations can generate novel and unexpected solutions, the pursuit of a superior solution requires navigating through many suboptimal mutations.

In this paper, we evaluate the strengths and weaknesses of alternative regulatory approaches in incentivizing a socially optimal balance between "exploration" and "exploitation" in evolutionary programming. The paper is structured as follows. In Section 2 we introduce the problem, focusing on the dual effects of evolutionary experimentation. First,

[3] Evolutionary algorithms are classified as metaheuristics, strategies that guide and adjust heuristics to find solutions beyond those achievable with standard local optimization techniques (Glover and Laguna, 1997). Several other metaheuristic algorithms draw inspiration from various natural processes ((Holland, 1992; Ghaheri et al., 2015). These include ant colony optimization (modeled after ant behavior; Dorigo & Stützle, 2004), artificial bee colony (based on bee behavior; Karaboga & Akay, 2009), Grey Wolf Optimizer (inspired by wolf pack dynamics; Mirjalili, Mirjalili, & Lewis, 2014), artificial neural networks (derived from neural systems; McCulloch & Pitts, 1943), simulated annealing (Kirkpatrick, Gelatt, & Vecchi, 1983), river formation dynamics (modeled after river creation; Rabanal, Rodríguez, & Rubio, 2007), and artificial immune systems (based on immune function; Dasgupta, 1999).

[4] External factors, new data, or evolving safety standards may necessitate ongoing adjustments, meaning what is deemed optimal at one point may become inadequate as conditions change. As a result, the self-programming algorithm may need to continue or be redesigned to adapt to these shifting goals.

we discuss the positive externalities that occur when superior mutations are identified, enhancing the performance of all subsequent uses of the algorithm. Second, we examine the negative externalities that arise when harmful mutations occur, raising complex legal and ethical questions about AI liability. In Section 3, we characterize the socially optimal level of evolutionary experimentation and consider the role of propertization and liability instruments in generating socially optimal incentives for evolutionary experimentation. Our results show that, to foster a socially optimal equilibrium, both types of externalities would need to be corrected, balancing the marginal risks and benefits for an optimal pace of technological progress. This could be accomplished by holding algorithm producers accountable for negative externalities from inferior mutations while allowing producers to capture the benefits from the discovery of superior mutations. An optimal balancing of these two externalities would encourage experimentation until the optimal stopping point and would prompt markets to reveal the comparative effectiveness of evolutionary programming compared to other programming techniques. Section 4 discusses the implications of our findings within the framework of current law.

2. Evolutionary Computation: Problems and Legal Instruments

Evolutionary computation traces its origins to the mid-20th century, inspired by the principles of Darwinian evolution and Mendelian genetics. Evolutionary and genetic programming are two subfields of evolutionary computation inspired by natural selection and biological evolution. Both methods rely on populations of candidate solutions, iterative improvement through random mutations, and selection of alternatives based on improved fitness.[5]

Mutation is a fundamental element in all these programming approaches, involving small, random changes that allow the search process for superior candidate solution to continue. The primary distinction between evolutionary and genetic programming lies in the use of

[5] Early foundational work includes Turing's concept of machine learning through a "genetical or evolutionary search" (Turing, 1950) and the introduction of evolutionary strategies by Rechenberg (1965) and Schwefel (1975), which emphasized optimization through natural selection and mutation. Holland (1975) formalized genetic algorithms, highlighting the role of crossover and selection in exploring solution spaces. For an overview, see Eiben and Smith (2015).

crossover. In evolutionary programming, mutation is the primary operator driving the evolution, while genetic programming employs mutation alongside crossover to enable adaptive changes and improvements to the program. Crossover is an operator that combines segments of the successful parent with the offspring solution. Simulating natural selection allows algorithms to improve iteratively by selecting the fittest candidates from a population of potential solutions, recombining and mixing their features, and introducing random mutations.[6]

Despite these differences, both methods employ mutations as a means of exploration, reflecting the fundamental dynamics of evolutionary processes. In both evolutionary approaches, mutations compete. The most effective variations continue to the next stage and the less efficient variations are discarded, akin to a "survival of the fittest" process, improving overall performance across the evolutionary process. As of today, companies like Volkswagen and Tesla use evolutionary algorithms to improve self-driving technologies. These manufacturers leverage the vast data collected from their fleet of vehicles to enhance their software.[7] Through this evolutionary process, algorithms adapt and improve their performance using real-world data. As the algorithm iterates, random mutations and crossovers may occasionally reduce performance to a level below the current state-of-the-art standard. In some cases, an inferior mutation could lead to a temporary loss in safety, financial returns, or productivity. However, when a superior mutation—such as improved obstacle detection in vehicles or increased efficiency in financial trades or automated manufacturing—is identified and adopted, this superior mutation creates a lasting advantage, enhancing the state-of-the-art benchmark for all future operations.

In the research phase of genetic algorithms, the optimal stopping point of this iterative process hinges upon the critical interplay between "exploration" and "exploitation." Genetic programming mutations produce both superior and inferior variants throughout the process. In

[6] Spector et al. (1999, vol. 3) describes the genetic programming process as one that "breed[s] programs over many generations, using the principles of natural selection, sexual recombination, and mutation." The successful programs replicate with slight modifications, refining their "genes" for subsequent tasks.

[7] This programming model is designed for continuous trial-and-error learning, allowing it to address emerging challenges. By receiving feedback data from the vehicles, the algorithm improves its ability to recognize patterns in driving environments. These advances are then deployed to all vehicles via updates to the manufacturer's software, such as Tesla's Full Self-Driving (FSD) software (Alwire, 2023).

the initial phase, the algorithm autonomously explores a vast search space, identifying solutions that might not have been anticipated by human designers, allowing the discovery of truly innovative approaches. As this iterative process progresses, the algorithm hones its focus to refine itself.[8]

2.1 Value of Experimentation: Superior Mutations and Propertization Instruments

As algorithms evolve through evolutionary programming, the discovery of mutations with higher fitness benefits all subsequent applications of the refined algorithms. Hereinafter, we shall refer to these improvements as "superior" mutations. Establishing appropriate legal frameworks to allocate the costs and benefits of evolutionary mutations will be essential for incentivizing continued innovation while promoting equitable allocations of the risks and gains produced by these technological advancements.

Patent applications for algorithms, especially in fields like artificial intelligence, machine learning, and data analytics, have surged in recent years. Once an algorithm reaches a satisfactory quality for adoption, the algorithm is generally patented for a specific application.[9] As it will be discussed in Section 4, while the value of the original program is captured by the manufacturer when the algorithm is patented and commercialized, the rapid pace of evolutionary programming presents issues for traditional patent systems. After the initial patent is obtained, the manufacturer's incentives to continue experimentation hinge upon the applicable propertization and liability mechanisms.

[8] The variance in fitness and mutation behavior in genetic algorithms has been well-documented in the field of evolutionary computing. For example, studies show that mutation rates play a critical role in preventing premature convergence and ensuring global optima are reached through ongoing exploration, even in later stages of algorithm evolution. As Goldberg and Rudnick (1991, p. 274) note, in this later phase, the refinement of an algorithm's population does not necessarily equate to reduced variance, particularly in non-uniform populations. For instance, while mutations in GPS technology for on-road self-driving cars generally result in fewer inferior variants over time, applying GPS technology to off-road self-driving introduces a less uniform search space, increasing mutation variance. This expanded search allows for greater exploration, leveraging a wider range of experimentation, which leads to a higher occurrence of both superior and inferior solutions.

[9] The application is a crucial component of a mutation's patentability—otherwise, a mutation runs the risk of being too abstract to be deemed patent-eligible Hasiguchi (2017).

2.2 Cost of Experimentation: Inferior Mutations and Liability Instruments

While mutations in evolutionary programming can yield superior solutions, they also generate variants that perform worse than their parent algorithms. This fact underscores the iterative nature of experimentation, where failure constitutes a crucial step toward success. The emphasis on learning from each failure reflects a growth mindset, which is vital in fields characterized by the value of the data gathered by random mutations. Failures in evolutionary experimental settings are not merely setbacks, but opportunities to validate models and uncover unforeseen challenges. Each mutation provides granular insights into system performance under real-world conditions, enabling an algorithm to identify flaws, optimize performance, and enhance reliability. In certain applications, the stakes are particularly high, given the inevitability of harmful mutations. We shall refer to these variants as "inferior" variants. Inferior variants are a necessary byproduct of the experimentation process that ultimately leads to evolutionary improvements of an algorithm. Evolutionary experimentation may additionally create harmful externalities. We shall refer to the subset of inferior mutations that causes losses as "harmful" mutations.

Problems of inferior and harmful mutations can be illustrated in the following two scenarios, both of which are characterized by heterogeneity of cases in the relevant search space. First, harmful mutations can persist in evolutionary algorithms when an algorithm overfits the training data, leading to exceptional performance within the narrow dataset but likely producing suboptimal results in new or unseen contexts (Sculley et al., 2018). These mutations may be selected for despite their limited practical utility beyond the specific conditions considered. To prevent overspecialization, cross-validation is essential to assess the fitness of evolving solutions in diverse environments. However, when cross-validation and generalization entails using an algorithm in real-world scenarios, individuals or entities may become inadvertent subjects of experimental validation, facing potential risks of harm or diminished performance while testing prospective updates or refinements.[10]

[10] Sculley et al. (2018) discuss the problem of overfitting in machine learning algorithms and highlight the risks of applying solutions optimized for specific datasets to broader, real-world contexts. The authors discuss the ethical challenges of deploying experimental models in diverse, real-life scenarios, aligning with our concerns about the risks faced by individuals during cross-validation.

A second class of harmful mutations arises at the opposite end of the probability distribution when evolutionary experimentation is carried out in real-life settings with heterogeneous environments. In these environments, a potentially harmful evolutionary mutation that outperformed a state-of-the art algorithm in observed applications could be adopted, even if it is unknowingly prone to catastrophic failure in unobserved cases (Amodei et al., 2016). In systems such as autonomous vehicles or financial modeling, this scenario poses significant danger because future deployment of such mutations could result in accidents or large financial losses. Additionally, undertaking rigorous testing of new mutants across a wide range of scenarios would be necessary to ensure robustness in evolutionary programming. Methods such as redundancy (evolving backup solutions such as rule-based overrides and human intervention) and robust fitness evaluation (evaluating solutions in diverse environments) might help safeguard against erratic outcomes (Amodei et al., 2016, pp. 14–15). Practically speaking, applying such backup and robustness methods would entail exposing a few individuals or entities to the risk of accidents or significant financial loss.

Preliminary statistical and econometric work has shown that robots outperform humans in improving their own programming. From a social welfare point of view, these programming techniques will lead to safer (and cheaper) technology over time. Humans defer to the superior skills of a robot and delegate important decisions to them (Casey, 2019). However, as robots increase their skills, their "thinking" becomes more "inscrutable," progressing beyond the human computational capacity (Michalski, 2018). Mulligan (2017) refers to these instances as "black box algorithms"—algorithms that not even their original designers and programmers can decipher. Machine learning and genetic and evolutionary programming have further increased the complexity and opacity of robots' decision-making processes. Given the opacity of a robot's decisions, it is difficult—and often unwise—for operators to second-guess and override the robot's decisions (Lemley and Casey, 2019). The high complexity of the decision algorithm and the dynamic adjustment of programming in unforeseen circumstances are what make robots different from other machines. According to many scholars, these characteristics also call for special legal treatment and a new approach to modeling accidents (Bertolini, 2014).

3. An Economic Model

In the following, we will develop an economic model to study four alternative legal regimes to evaluate their strengths and weaknesses in incentivizing the adoption of evolutionary programming technologies and optimal exploration of evolutionary mutations. In Section 3.1, we set the stage identifying the socially optimal level of evolutionary experimentation. In Sections 3.2 through 3.5, we will consider how the alternative legal regimes under consideration align manufacturers' private incentives to the socially optimal ideal.

3.1 Socially Optimal Experimentation

Consider a scenario where an algorithm has been proven fit in laboratory testing with the ability to carry out an assigned task with a minimal error rate, e. The algorithm is adopted for commercial production as the state-of-the-art algorithm in the industry, surpassing the safety level that humans would achieve in carrying out that activity. Real-world testing confirms the fitness of the algorithm and consumers report errors in the limited range predicted in the laboratory testing.

The manufacturer faces the opportunity to engage in evolutionary programming refinements of the algorithm, evaluating mutations of the state-of-the-art program in real-world testing. This means that each mutation would be applied in a practical, operational environment. A mutation will be deemed inferior if its performance is lower than what could have been achieved by the current state-of-the-art version of the algorithm. A subset of inferior mutations (harmful mutations) could lead to a loss due to the lack of fitness of the resulting algorithm. Conversely, a mutation will be deemed superior if its performance surpasses that of the existing algorithm. Once identified, a superior mutation is adopted and, going forward, all subsequent operations benefit from the superior algorithm.

The choice faced by the manufacturer underscores the delicate trade-off between risk and reward inherent in the evolutionary process of algorithms. Each time a new mutation is tested, there is a potential cost in the form of a one-time performance loss if the mutation proves harmful and underperforms the state-of-the-art algorithm. The mutation's underperformance may cause a financial loss. Conversely, successful mutations carry the promise of long-term,

cumulative rewards, as they improve the algorithm's performance in present applications and in all future operations. These beneficial mutations establish new benchmarks, raising the standard against which future mutations will be judged. The ongoing challenge lies in navigating this trade-off: minimizing the short-term risks of harmful mutations while capitalizing on the long-term benefits of beneficial mutations, thereby optimizing the evolutionary path of the algorithm.

In evolutionary programming, mutations play a crucial role in the early stages by helping to explore a broad range of possible solutions and potentially driving improvements. However, as the algorithm progresses, the likelihood that random mutations will enhance the current solution diminishes. At this stage, most mutations are more likely to result in inferior solutions. The socially optimal stopping point in an evolutionary program's refinement can be identified by balancing the costs associated with underperforming mutations against the benefits of successful mutations. Let R represent the long-term cumulative reward from a successful mutation; p be the probability of a mutation being "superior"; and $(1-p)$ be the probability of a mutation being "inferior" relative to a current state-of-the art algorithm. With probability γ, the mutation is "harmful" and causes harm H to user, such that γH represents the expected one-time loss caused by a harmful mutation. $C(n)$ is the private cost of each mutation.

Consider a proposed round of evolutionary variations with n mutations tested.[11] The expected value of the evolutionary process at any given iteration can be represented by a function $E(n)$, where:

$$E(n) = p^n R(n) - n(1-p)\gamma H - C(n)$$

To find the socially optimal stopping point, we need to maximize the expected value of $E(n)$ over the course of the algorithm's refinement process. The algorithm should continue evolving until the expected gain from an additional mutation is less than or equal to zero, at which point the cost of risk outweighs the potential reward. Therefore, the optimal stopping point is when:

[11] Without loss of generality, our model mirrors the stylized functioning of evolutionary programming, where inferior mutations are discarded and superior mutations replace the current program, serving as benchmarks for subsequent mutation rounds. Similar qualitative outcomes would emerge under genetic programming, where the selection process would incorporate the crossover of successful parent and offspring solutions.

$$\frac{dE(n)}{dn} = 0$$

Analytically, the optimal stopping point provides the solution to the following equation:

$$p^n R' + lnp\ p^n R = (1 - P)\gamma H + C'$$

The equation defines the socially optimal balance in evolutionary programming between continued exploration and exploitation to mitigate the risk of detrimental mutations that deviate from the state-of-the-art program. The optimal stopping point is achieved when the probability-adjusted rewards equal the risk-adjusted costs. This equilibrium maximizes the net prospective gains from improvements, accounting for the risks posed by harmful mutations.

Rearranging terms in the above equation, we get:

$$P^n R' = (1 - P)\gamma H - lnPP^n R + C'$$

where the LHS can be interpreted as the marginal expected benefit from evolutionary programming given by the increase in long-term cumulative reward from the implementation of the n^{th} successful evolutionary mutation times the probability of implementing it. Meanwhile the RHS measures the expected marginal cost of experimentation of the n^{th} round of experimentation. The RHS is the sum of two costs: the probability of failure $(1 - p)$ multiplied by the harm that a harmful mutation γH may cause, and the reduction in probability of a successful mutation $(lnPP^n)$ in obtaining the benefit R. Evolutionary experimentation should proceed until the marginal expected benefit of experimentation is greater than its expected costs. Testing mutations should stop when these marginal expected benefits and costs are equal.

3.2 Privately Optimal Experimentation

Private incentives to continue experimentation may not always align with the socially optimal stopping point. Unlike socially optimal values, private incentives are affected by the

legal rules that govern two critical factors of evolutionary programming: (1) the producers' ability to capture the added value generated by superior mutations and (2) the allocation of losses caused by harmful mutations. Intellectual property and liability rules will be critical in determining these two factors and in deciding the extent to which producers internalize the benefits of continued innovation while simultaneously influencing the risks from experiments resulting in negative externalities. This dynamic can lead to either over-experimentation, where producers pursue additional mutations beyond the socially optimal point, or under-experimentation, where liability may deter desirable innovation.

Producers' valuation of the evolutionary process at any given iteration can be represented by a function $\pi(n)$

$$\pi(n) = p^n \delta R(n) - C(n) - n(1-p)\gamma\alpha D$$

Producers' expected return $\pi(n)$ depends on three key factors: the value of the evolutionary programming $R(n)$, the production cost $C(n)$, and the liability for losses caused by harmful mutations, αD. The parameter δ represents the producers' ability to capture long-term cumulative returns from successful mutations, with $\delta = 1$ indicating full appropriation of returns and $\delta = 0$ reflecting an inability to capture any returns from superior mutations. Similarly, when the liquidated damages determined by a court is the harm H caused by a harmful mutation (D=H), the parameter α reflects the extent to which the manufacturer internalizes losses from harmful mutations, where $\alpha = 1$ signifies full liability and $\alpha = 0$ denotes no liability.

Evolutionary experimentation should proceed until the expected gain from an additional mutation equals zero. At this point, the risks posed by harmful mutations outweigh the potential benefits of superior mutations. Thus, the optimal stopping point is defined as:

$$\frac{d\pi(n)}{dn} = 0$$

namely

$$\delta P^n R' = C' + (1-P)\gamma\alpha D - \delta lnP P^n R$$

where the LHS represents the marginal expected benefit of experimentation, calculated as the increase in the long-term cumulative reward from implementing the n^{th} successful evolutionary mutation, multiplied by the probability of its successful implementation. The RHS reflects the risk associated with the n^{th} round of experimentation.

3.3 Aligning Incentives Through Propertization and Liability Mechanisms

The interplay between propertization and liability mechanisms is pivotal in aligning private incentives with the socially optimal level of evolutionary experimentation. While evolutionary programming offers significant potential for technological advancement, the risks associated with harmful mutations and the challenges of capturing the downstream value of superior mutations create a need for carefully calibrated legal frameworks. In the following sections, we examine how alternative combinations of propertization and liability instruments influence the incentives of producers and their alignment with social objectives.

We analyze four distinct legal regimes reflecting different degrees of propertization and liability for algorithmic outcomes. These regimes range from laissez-faire approaches, where neither the benefits nor the harms of mutations are internalized, to systems that couple the propertization of the benefits from superior mutations with the liability arising from harmful mutations.[12] By exploring these frameworks, we illustrate the economic tradeoffs of evolutionary programming.

[12] The use of propertization and liability instruments would clearly face implementation challenges in cases involving open-source algorithms or collaborative frameworks.

		Propertization of Superior Mutations	
		No $\delta = 0$	Yes $\delta = 1$
Liability for Harmful Mutations	No $\alpha = 0$	Laissez-Faire	Propertization without Liability
	Yes $\alpha = 1$	Liability without Propertization	Liability & Propertization

Table 1: *Four legal regimes*

We will now proceed to analyze the incentives created by the four regimes illustrated in Table 1, characterized by different combinations of propertization and liability.

Laissez-Faire Regime ($\delta = 0$ and $\alpha = 0$)

In a regime characterized by $\delta = 0$ and $\alpha = 0$ (referred to here as "Laissez-Faire" regime), producers are unable to capture the returns from superior mutations in evolutionary programming. Simultaneously, they bear no liability for harmful mutations. Consequently, the producer internalizes neither the potential gains from innovation nor the repercussions of failure. This results in an optimal stopping point governed by the following condition:

$$C' = 0$$

namely, when the private marginal cost of producing the n^{th} mutation and the expected cost of liability are zero. Although producers face no liability for harmful mutations ($\alpha = 0$), they will still underinvest in evolutionary programming because they cannot capture the long-term value of future algorithmic mutations ($\delta = 0$).

Propertization without Liability ($\delta = 1$ and $\alpha = 0$)

In a regime characterized by $\delta = 1$ and $\alpha = 0$ (referred to here as "Propertization without Liability" regime), producers can capture the returns from superior mutations in the evolutionary programming. This boosts their incentives to invest in innovation and pursue superior algorithmic improvements, as producers stand to gain the benefits of their advances. Producers can avoid liability for harmful mutations, externalizing the losses caused by failed experimentation. This results in a stopping point governed by the following condition:

$$P^n R' = C' - lnPP^n R$$

Hence, the private marginal cost of evolutionary programming will be lower than the social marginal cost, since producers are able to externalize losses caused by harmful mutations. Producers will thus overinvest in experimentation.

Liability without Propertization ($\delta = 0$ and $\alpha = 1$)

In a regime characterized by $\delta = 0$ and $\alpha = 1$ (referred to here as "Liability without Propertization" regime), producers cannot capture the returns from superior mutations in evolutionary programming while bearing full liability for the harmful mutations. This results in a stopping point governed by the following condition:

$$C' + (1 - p)\gamma H = 0$$

Hence, the private marginal cost of evolutionary programming will be higher than the private marginal benefit, and producers will underinvest in experimentation, compared to the social optimum.

Liability & Propertization ($\delta = 1$ and $\alpha = 1$)

In a regime characterized by $\delta = 1$ and $\alpha = 1$ (referred to here as "Liability & Propertization" regime), producers can fully appropriate the cumulative long-term value of successful

mutations from evolutionary programming. Producers also face full liability for the harmful mutations. The stopping point in this regime reflects a tradeoff between the expected returns from superior mutations and the anticipated liability from harmful mutations:

$$P^n R' = C' + (1 - P)\gamma H - lnPP^n R$$

Under this regime, since $\delta = 1$ and $\alpha = 1$, producers operate within a balanced framework where they internalize both the benefits and negative externalities of evolutionary experimentation. Hence, the private marginal cost of evolutionary programming will be equal to the social marginal cost, thereby fostering incentive alignment. Producers will engage in the socially optimal level of evolutionary experimentation.

3.4 Comparison of Regimes

In Figure 1 below, we illustrate how the interplay of propertization and liability regimes, namely the parameters δ and α, influence optimal experimentation. The socially optimal level of experimentation, n^{**} (and the resulting optimal stopping point) is reached when the social marginal benefit and marginal cost of experimentation equal one another, as represented by the solid (black) line.

Regimes of full propertization and no liability (or incomplete liability), $\delta = 1$ and $0 \leq \alpha < 1$, will generate excessive experimentation, ${n_\delta}^* > n^{**}$, as represented by the two (light and dark blue) rightmost curves of Figure 1. In the limiting case with no liability and zero marginal cost of experimentation, $\alpha = 0$ and $C' = 0$, experimentation will continue indefinitely, ${n_\delta}^* = \infty$, as represented by the far right (dark blue) dotted curve. Excessive levels of experimentation may also occur in cases of partial liability $0 < \alpha < 1$ and for positive marginal cost of experimentation, $C' > \delta R' - (1 - p)\gamma H$, as represented by the second from right (light blue) dotted curve.

Regimes of full liability and no propertization (or incomplete propertization), $\alpha = 1$ and $0 \leq \delta < 1$, will generate suboptimal levels of experimentation, ${n_\alpha}^* < n^{**}$, as represented by the two (orange and red) leftmost curves of Figure 1. In the limiting case with no propertization, $\delta = 0$, no experimentation occurs for any values $C' \geq 0$ and $\alpha \geq 0$, as

represented by the red dashed line on the far left of Figure 1. The absence of experimentation, $n_\alpha{}^* = 0$, is intuitive since firms cannot appropriate any of the gains that result. In the intermediate range of cases where both liability and propertization are present, experimentation will take place at a suboptimal level, $n_\alpha{}^* < n^{**}$ when liability exceeds propertization, $0 < \delta < \alpha \leq 1$, as represented by the second from left (orange) dashed curve in Figure 1.

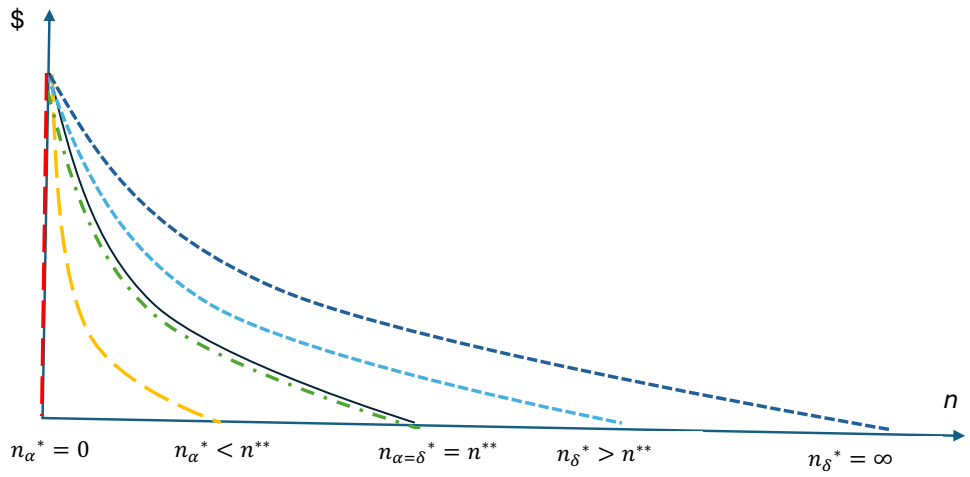

Figure 1: *Level of Experimentation under Different Regimes*

Under a regime of full propertization and full liability ($\delta = 1$ and $\alpha = 1$), private incentives align with social optimality. Experimentation will be carried out until the socially optimal stopping point, $n_{\alpha=\delta}{}^* = n^{**}$, as illustrated by the green curve overlapping the solid (black) curve, which represents the socially optimal condition. When the cost of experimentation $C' = 0$, intermediate regimes ($\delta < 1$ and $\alpha < 1$) can also yield socially optimal incentives, provided that $\frac{\delta}{\alpha}$ varies proportionally. Specifically, when $C' = 0$, a perfectly balanced regime of partial liability and partial propertization, $0 < \delta = \alpha < 1$ (i.e., $\frac{\delta}{\alpha} = 1$) will incentivize firms to undertake a socially optimal level of experimentation, $n_{\alpha=\delta}{}^* = n^{**}$. When $C' > 0$, there will also exist an optimal ratio of propertization and liability that can align private

and social experimentation incentives. However, the alignment of private and social incentives necessitates a reduction of liability relative to propertization, $\frac{\delta}{\alpha} > 1$.[13]

4. Discussion

To achieve a socially optimal equilibrium n^{**}, it is essential to internalize all externalities, both positive and negative. Propertization and liability instruments are critical tools for designing effective incentive mechanisms for evolutionary programming. Our analysis highlights that the combined use of these instruments enables producers to internalize the costs of experimentation while capturing the downstream benefits of successful innovations. This dual internalization ensures the optimal use of evolutionary computation, allowing experimentation to proceed until the optimal stopping point is achieved.

The internalization of gains and losses from algorithmic innovation further provides a market mechanism to determine the financial attractiveness of evolutionary programming compared to more traditional programming methods that do not entail field experimentation. In equilibrium, the producers' choices reflect the social desirability of evolutionary programming within the broader context of computational methodologies.

4.1 Implementing Propertization

Evolutionary computation has been utilized industrially since the 1980s. In 1988, John Koza patented the first genetic algorithm. Important academic research highlighted the growing recognition and applications of genetic programming across various fields over the last twenty years (Banzhaf et al., 1998 and Katoch et al., 2020).

Slowik and Kwasnicka (2020) underscore the critical role of mutation in the evolutionary search process. The small, random changes that enable the search for improved solutions often pose challenges in securing intellectual property protection. Under current law, several obstacles complicate the patenting of subsequent evolutionary mutations of algorithms. First, for a mutation to qualify for a patent update, it must represent a substantial improvement, yet many evolutionary changes offer only marginal gains. Second, the rapid pace of

[13] See proof in the Appendix.

evolutionary experimentation can outpace the patent process, with notable mutations occurring before an update can be filed.[14] Finally, attempts to patent new mutations often reveal overlaps with the original patent—most mutations fail to meet novelty and non-obviousness standards because mutations are, by their own nature, derivative adaptations of existing algorithms.[15] The challenge of protecting small, incremental changes through patents was addressed in the WIPO Intellectual Property Guide (2020). This publication discusses uncertainty regarding the scope of patent claims when applied to a generic function without a specific structure, which may also be relevant to the minor, random alterations introduced by mutation operators in an evolving algorithm.[16]

In short, the current patent system was not designed to accommodate the rapid, iterative nature of evolutionary programming, making it difficult to keep pace with the progress of these technologies. While algorithms may evolve quickly, patenting their progression remains a challenge. Firms investing in evolutionary programming may thus struggle to capture the full value of their innovations. As a result of these implementation difficulties, the cumulative value of the evolutionary mutations may be under-protected by the traditional patent framework, leading to imperfect propertization, potentially disincentivizing innovation in fields reliant on iterative, mutation-driven optimization.

---

[14] Beyond these administrative challenges, accelerated innovation processes could call into question whether a patent's traditional twenty-year period of exclusivity is too long for the balance of incentivizing disclosure of innovation without excessively hampering competitors' innovation. *See* Giacobbe (2022).

[15] In addition to the novelty of the mutation itself, the use of an algorithm or inventive machine to generate that mutation potentially complicates the obviousness inquiry. Whereas traditionally, the U.S. Trademark and Patent Office would evaluate obviousness based on what a hypothetical, non-inventive, skilled person would find obvious, evolutionary mutations might need to be evaluated against what a hypothetical algorithm might find obvious. See Abbot (2019).

Another potential barrier to patenting successful mutations is the question of whether the manufacturer of an algorithm or the algorithm itself qualifies as the "inventor" of the mutation, since all patents require an inventor. Whereas, in *Thaler v. Vidal*, 43 F.4th 1207 (4th Cir. 2022), the Fourth Circuit held that artificial intelligence cannot be an "inventor" for purposes of a patent, whether a human can be the "inventor" of an innovation devised by artificial intelligence remains an open question. For a summary of scholarly views on this topic, see Wu (2023).

[16] Traditional patent protection may not always suffice for the small, iterative improvements typical of evolutionary algorithms. The small scale of individual mutations may lack the novelty or inventive step required for patentability On a related issue, the Max Planck Institute for Innovation and Competition highlighted the inadequacy of the current patent system for protecting new genomic techniques (NGTs) in its position statement (Kim et al. 2024).

4.2 Implementing Liability

As discussed in Guerra et al. (2021a and 2021b), the negative externalities that arise in the process of evolutionary experimentation raise the question of how tort or product liability rules should handle liability in cases where autonomous systems malfunction due to evolutionary changes in their code. The dynamic adjustment of programming to unforeseen circumstances has posed challenges to policymakers and tort scholars alike, setting these systems apart from traditional programming methods. With these programming techniques, attributing responsibility for robot accidents to a specific party is increasingly difficult. This problem has been referred to as the "responsibility gap" (Mathias, 2004).[17] Policymakers and legal scholars fear that applying existing legal rules to robot accidents may slow down desirable experimentation with these new programming techniques. Specifically, policymakers are concerned that imposing liability on manufactures (and/or their programmers/employees) would disincentivize employment of these new programming techniques. Likewise, policymakers fear that shifting liability for accidents caused by "defective genes" onto robot owners would undermine prospective purchasers' incentives to buy these novel robots, effectively driving these programming techniques out of the market.

Significant ethical and legal challenges arise when evolutionary experimentation is conducted "in the field," especially when third parties become passive subjects of the evolutionary programming process. Ethical concerns center on the need for informed consent and transparency, given the natural unpredictability of random mutations. In the following, we will explore how an economic approach to this issue can embrace these ethical considerations.

For the mechanisms we propose to function effectively, producers must provide full disclosure and ensure that participants are willing to participate in field experimentation, with clear disclosure of the associated risks, benefits, and uncertainties. Participants should have the freedom to opt in or out based on this information. Producers, in turn, will develop compensation schemes to secure the required level of voluntary participation. Ensuring informed and voluntary participation not only upholds ethical and legal standards but also

[17] Matthias (2004) formulated the concept of "responsibility gap" when considering the responsibility for harm caused by "learning automata" (i.e., machines that learn from experience, rather than being explicitly programmed). See also the recent discussion by Oimann and Tollon (2024).

maintains public trust and safety. To achieve these objectives, three potential compensation models warrant consideration:

1. *Ex Ante Compensation*: Under this option, individuals could opt to participate in field experimentation in exchange for some form of compensation for the incremental risks they face from exposure to potential harmful mutations, regardless of whether any actual harm occurs. Evolutionary algorithm programming causes an ex ante expected harm equal to $\sum_{i=1}^{n^{**}} P^{n^{**}-i}(1-P)^{i}\gamma H$ when socially optimal experimentation level n^{**}, is adopted. This ex ante approach acknowledges the participants' role in the experimentation process and provides upfront payment for assuming the risks of evolutionary programming. By offering compensation before experimentation begins, this model not only encourages participation but also operates as a proactive means of internalizing the costs associated with evolutionary experimentation.

2. *Ex Post Compensation*: Under this option, individuals who choose to participate receive ex post compensation for the harm caused by a harmful mutation. Compensation in this case is calibrated based on the incremental risk that participants faced from the inferior performance of the mutation relative to the state-of-the-art algorithm, and is equal to H. The ex post compensation model ensures that producers face the external costs of their experimentation. Unlike the ex ante compensation option, this approach limits producers' financial outlays to instances of realized harm.

3. *Hybrid Approach*: A hybrid alternative to the two options considered above could combine the ex ante and ex post elements, offering a smaller initial payment to incentivize participation with conditional compensation should harm occur. This approach balances the need to incentivize participation with a fair balance of compensation in the event of negative outcomes. The initial payment helps to incentivize entry for participants, while compensation for the harm is conditional upon the occurrence of harm that is caused by a harmful mutation. If properly calibrated, this hybrid solution can ensure voluntary participation in evolutionary research with an ex ante payment, mitigate financial risk for participants with ex post compensation, and in turn lead producers to fully internalize the negative externalities produced by harmful mutations.

Legal challenges naturally arise when implementing liability instruments for harm caused by harmful mutations. Establishing causation poses a potential problem for implementing liability instruments. Evolutionary computation programs rely on stochastic processes, making it difficult to trace a direct causal link between a specific mutation and the resulting harm. Further, the outcomes of evolutionary programs typically depend on multiple interacting factors, including intervening actions by users, third-party software applications, and environmental conditions. The interplay between hardware, software, human, and other intervening environmental factors makes isolating the effect of a harmful mutation as the cause of the harm a difficult exercise.

Quantifying compensable harm also presents significant challenges. Evolutionary algorithms often operate with probabilistic outcomes, making it difficult to predict what would have occurred in the absence of a harmful mutation. Such counterfactual scenarios are by their own nature speculative, and courts may hesitate to accept them as a basis for awarding damages. Moreover, some harmful mutations may produce undesirable results that fall within the expected risks of using evolutionary computation, thereby evading the scope of products liability law.[18]

Additional evidentiary complexities arise in the implementation of liability, given the complexity of the data needed to establish liability for evolutionary computation processes. Developers, who typically have exclusive access to proprietary information regarding algorithmic design and the selection of mutations within their processes, should have an obligation to mitigate these evidentiary difficulties by maintaining comprehensive and intelligible records of algorithmic mutations. Ultimately, resolving these legal challenges would require courts to attain a nuanced understanding of evolutionary computation, coupled with legal standards that account for the new reality put forth by these programming technologies.

Notwithstanding these implementation difficulties, each of the compensation models discussed above, combined with an appropriate propertization mechanism, can foster

[18] Applying products liability frameworks to evolutionary algorithms would require distinguishing between outcomes that are acceptable and those that are legally actionable. Determining whether the harm that stems from an inferior mutation falls within the range of tolerable risk compared to the alternative state-of-the-art algorithmic design may compel courts to develop new liability standards specific to evolutionary computation. On this point, see the liability standard proposed by Guerra et al. (2023).

environments that sustain an optimal level of evolutionary programming. By carefully allocating the costs and benefits of experimentation, these mechanisms safeguard the voluntariness of participation, while requiring producers to internalize both the positive and negative externalities associated with evolutionary programming. These approaches would drive experimentation until the optimal stopping point, achieving a socially optimal level of adoption of evolutionary computation techniques.

Given the shortcomings of existing intellectual property frameworks in securing property rights over the benefits of evolutionary processes, it is unsurprising that evolutionary computation techniques are predominantly employed by large firms (e.g., Tesla and Volkswagen). These companies can better internalize the downstream benefits of superior mutations even in the absence of effective propertization mechanisms, while also managing the risks of harmful mutations through portfolio diversification. In contrast, smaller firms in the industry or those with shorter time horizons (e.g., start-up companies) face greater challenges in capturing the benefits of experimentation and in distributing liability risks across their operations. In the absence of well-balanced propertization and liability regimes, limiting liability may function as a second-best mechanism to offset the constraints of intellectual property protection and incentivize investment in evolutionary research also for smaller firms in the industry. By reducing the risk borne by innovators, such an approach can encourage optimal levels of experimentation, leading to socially desirable financial returns from potentially transformative evolutionary programming methods. However, the possibility of harmful experimentation raises concerns about fairness in risk allocation, necessitating carefully designed complementary safeguards to balance limited liability and research incentives with broader social and economic interests.

References

Abbott, R. (2019). Everything is Obvious. UCLA Law Review, 66, 2.

AIwire. (2023, March 8). How Tesla uses and improves its AI for autonomous driving. https://www.aiwire.net/2023/03/08/how-tesla-uses-and-improves-its-ai-for-autonomous-driving/.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mané, D. (2016). Concrete Problems in AI Safety. arXiv. https://doi.org/10.48550/arXiv.1606.06565.

Banzhaf, W., Nordin, P., Keller, R. E., & Francone, F. D. (1998). Genetic Programming: An Introduction. Morgan Kaufmann Publishers.

Bertolini, A. (2014). Robots and liability: Justifying a change in perspective. Pisa University Press.

Bertolini, A. (2020). Artificial intelligence and civil liability. European Parliament – Committee on Legal Affairs.

Bertolini, A., & Riccaboni, M. (2020). Grounding the case for a European approach to the regulation of automated driving: The technology-selection effect of liability rules. European Journal of Law and Economics, 1-42.

Casey, B. (2019). Robot ipsa loquitur. Georgetown Law Journal, 108, 225.

Chopra, S., & White, L. F. (2011). A legal theory for autonomous artificial agents. University of Michigan Press.

De Chiara, A., Elizalde, I., Manna, E., & Segura-Moreiras, A. (2021). Car accidents in the age of robots. International Review of Law and Economics, 106022.

Eiben, A. E., & Smith, J. E. (2015). Introduction to evolutionary computing (2nd ed.). Springer.

European Parliament. (2017). European Parliament resolution of 16 February 2017 with recommendations to the Commission on Civil Law Rules on Robotics (2015/2103-INL). European Parliament.

Ghaheri, A., Shoar, S., Naderan, M., & Hoseini, S. S. (2015). The applications of genetic algorithms in medicine. Oman Medical Journal, 30(6), 406–416. https://doi.org/10.5001/omj.2015.82.

Giacobbe, T. (2022). Adapting to Challenges Posed by the Fourth Industrial Revolution: A Regulatory Call to Action Concerning Cybernetic Technology. Washington University Jurisprudence Review, 15, 141.

Glover, F., & Laguna, M. (1997). Tabu search. Springer Science & Business Media.

Goldberg, D. E., & Rudnick, M. (1991). Genetic algorithms and the variance of fitness. Complex Systems, 5, 265-278.

Guerra, A., Parisi, F., & Pi, D. (2021a). Liability for robots I: Legal challenges. Journal of Institutional Economics.

Guerra, A., Parisi, F., & Pi, D. (2021b). Liability for robots II: An economic analysis. Journal of Institutional Economics.

Hasiguchi, M. (2017). The Global Artificial Intelligence Revolution Challenges Patent Eligibility Laws. Journal of Business & Technology Law, 13, 1.

Hubbard, F. P. (2014). Sophisticated robots: Balancing liability, regulation, and innovation. Florida Law Review, 66, 1803.

Kassahun, Y., Yu, B., Tibebu, A. T., Stoyanov, D., Giannarou, S., Metzen, J. H., & Vander Poorten, E. (2016). Surgical robotics beyond enhanced dexterity instrumentation: A survey of machine learning techniques and their role in intelligent and autonomous surgical actions. International Journal of Computer Assisted Radiology and Surgery, 11, 553–568.

Katoch, S., Chauhan, S. S., & Kumar, V. (2020). A review on genetic algorithm: past, present, and future. Multimedia Tools and Applications, 80(5), 8091-8126. https://doi.org/10.1007/s11042-020-10139-6

Kim, D., Kock, M. A., Lamping, M., Batista, P. H. D., Hilty, R. M., Slowinski, P. R., & Steinhart, M. (2024). New genomic techniques and intellectual property law: Challenges and solutions for the plant breeding sector – Position statement of the Max Planck Institute for Innovation and Competition: Munich, 8 January 2024. GRUR International, 73(4), 323–339. https://doi.org/10.1093/grurint/ikae017.

Lemley, M. A., & Casey, B. (2019). Remedies for robots. The University of Chicago Law Review, 86, 1311–1396.

Matthias, A. (2004). The responsibility gap: Ascribing responsibility for the actions of learning automata. Ethics and Information Technology, 6(3), 175-183.

Michalski, R. (2018). How to sue a robot. Utah Law Review, 2018(5), 1021-1074.

Mulligan, C. (2017). Revenge against robots. South Carolina Law Review, 69, 579-598.

Oimann, A.K., & Tollon, F. (2024). Responsibility gaps and technology: Old wine in new bottles? Journal of Applied Philosophy. https://doi.org/10.1111/japp.12763.

Rechenberg, I. (1965). Cybernetic solution path of an experimental problem. Royal Aircraft Establishment.

Schwefel, H.-P. (1975). Evolution and optimum seeking. Wiley-Interscience.

Sculley, D., Snoek, J., Wiltschko, A., & Rahimi, A. (2018). Winner's Curse? On the Optimality of Solutions and Hyperparameter Optimization. arXiv preprint arXiv:1811.06461.

Slowik, A. and Kwasnicka, H. (2020). Evolutionary algorithms and their applications to engineering problems. Neural Computing and Applications, 32, 12363–12379. https://doi.org/10.1007/s00521-020-04832-8.

Spector, L., Langdon, W.B., O'Reilly, U., and Angeline, P.J. (1999). Advances in Genetic Programming, Volume 3 (1999).

Holland, J. H. (1975). Adaptation in natural and artificial systems. University of Michigan Press.

Turing, A. M. (1950). Computing machinery and intelligence. Mind, LIX (236), 433–460.

World Intellectual Property Organization. (2020). Intellectual property guide for genetic resources and genetic sequence data: Integrated intellectual property management for genetic material and genetic sequence data. https://www.wipo.int/export/sites/www/tk/en/docs/ip_gr_grdataguidefin.pdf.

Wu, J. (2023). Bridging the AI Inventorship Gap. Fordham Law Review, 91, 2515-2547.

Appendix

The socially optimal condition SOC is:

$$p^n(R' - nlnpR) = (1-p)\gamma H + C'$$

The privately optimal condition POC is:

$$\delta p^n(R' - nlnpR) = (1-p)\gamma\alpha D + C'$$

It is immediate to observe that when $\delta = \alpha = 1$ and $D = H$ (perfect damage assessment by courts), SOC and POC coincide and $n^* = n^{**}$. Taking the ratio LHS and RHS for both equations, we can rewrite SOC and POC respectively as:

$$\frac{(1-p)\gamma H}{p^n(R' - nlnpR)} + \frac{C'}{p^n(R' - nlnpR)} = 1$$

$$\frac{(1-p)\gamma\alpha D}{\delta p^n(R' - nlnpR)} + \frac{C'}{\delta p^n(R' - nlnpR)} = 1$$

By comparing the two equations above, SOC and POC also coincide when $\frac{\delta}{\alpha} = 1$, when the marginal cost of experimentation is zero, $C' = 0$, i.e. a balanced regime of imperfect liability and propertization, $0 < \delta = \alpha < 1$, will produce n^{**}. When $C' > 0$,

$\frac{C'}{\delta p^n(R'-nlnpR)} > \frac{C'}{p^n(R'-nlnpR)}$ since $\delta < 1$, which implies that $\frac{\delta}{\alpha} > 1$ for equating POC and SOC. This implies that there exists an optimal ratio of propertization and liability, $\frac{\delta}{\alpha} > 1$, that can align private and social experimentation incentives. In presence of high marginal cost of experimentation, namely when $C' > \delta R' - (1-p)\gamma H$, it will always be optimal to set experimentation n to zero.