When Evolution Cannot Evolve: Diagnosing Disconnected Genotype-Phenotype Mappings in Autonomous AI Development Systems

Vasyl Golubenko
TOV ZELTREX, Zhytomyr, Ukraine
vasyl.golubenko@zeltrex.com

March 2026

Abstract

Evolutionary optimization is a promising approach for meta-tuning autonomous AI development systems, yet its effectiveness depends on a causal link between evolved parameters and task outcomes. This paper reports a production failure mode in which 477 genome mutations across 28 generations produced a net fitness decline of 7.4%, with no individual mutation yielding measurable improvement. Five diagnostic experiments confirm that the genotype-phenotype mapping is entirely severed: four of six genome genes are never applied to execution, the remaining two are overridden by downstream components, and fitness variance is explained by task category rather than genome parameters (between-niche variance accounts for the majority of fitness differences, while within-niche standard deviations range from 0.065 to 0.250). This failure mode, termed *Disconnected Evolution*, is formalized as a specific manifestation of Goodhart's Law [15] in which the optimization target is causally independent of the optimization variables. A taxonomy of eight disconnection mechanisms is developed from the evolutionary computation, neural architecture search, meta-learning, and prompt optimization literatures. Detection methods (generation trend analysis, elite-population comparison, parameter-fitness correlation, niche decomposition) and architectural remedies (gene-execution wiring, niche-normalized fitness, quality-diversity archives) are proposed. Among known autonomous AI agents, only two employ evolutionary optimization, and both exhibit disconnection pathologies, suggesting that validated genotype-phenotype mappings are a prerequisite for evolutionary meta-optimization in this domain.

Keywords:

1 I. Introduction

Autonomous AI development systems that generate, execute, and evaluate tasks without human intervention face a meta-optimization challenge: tuning the parameters that govern the AI agent's behavior over time. Evolutionary algorithms are a natural candidate for this meta-optimization because they maintain populations of configurations (genomes), evaluate them against real-world outcomes (fitness), and breed improved configurations through selection, crossover, and mutation [20].

Night Shift is an autonomous development system that has operated continuously for 21 days, completing over 500 software engineering tasks. Its evolutionary engine implements NSGA-II selection [5] with island models, adaptive mutation rates following the Rechenberg 1/5 rule, and textual gradients generated by a language model. The system maintains a population of 20 genomes per generation, each encoding six parameters: model selection, token budget, prompt style, context depth, continuation limit, and task scope. After 28 generations of evolution, the system accumulated 564 evaluated genomes and 477 tracked mutations.

None of the 477 mutations improved fitness.

This outcome is not attributable to algorithmic deficiency. The evolutionary operators (NSGA-II, crossover, adaptive mutation) are correctly implemented and well-established in the literature [5, 20]. Rather, the failure arises from a complete disconnection between the evolved genotype and the expressed phenotype: genome parameters do not causally influence task execution or its evaluation. This paper presents a systematic diagnosis of this disconnection, surveys related phenomena across multiple optimization domains, and proposes both detection methods and architectural remedies.

A companion paper addresses the model routing subsystem that contributes to the disconnection: Golubenko, V. (2026). Output-Type-Aware Model Routing in Autonomous Development Systems. TOV ZELTREX Technical Report.

The contributions of this paper are as follows:

1. Documentation and formalization of Disconnected Evolution as a failure mode in production autonomous AI systems.
2. Five diagnostic experiments quantifying the disconnection using production data from 564 genome evaluations.
3. A taxonomy of eight disconnection mechanisms synthesized from evolutionary computation, NAS, meta-learning, and prompt optimization literatures.
4. Detection methods and architectural remedies for restoring genotype-phenotype causality.

2 II. Related Work

This section identifies eight mechanisms by which optimization can become disconnected from outcomes, each documented in prior work.

2.1 A. Goodhart's Law and Proxy Optimization

Manheim and Garrabrant [15] taxonomize four variants of Goodhart's Law. The failure mode reported here corresponds to the *causal* variant: genome parameters correlate with fitness during initial observation but do not cause fitness changes. Intervening on the genome through mutation fails to affect outcomes because the causal path is severed by downstream overrides.

Gao et al. [8] formalize proxy-target divergence for reward model overoptimization in RLHF, demonstrating that proxy reward increases monotonically while true reward follows an inverted-U curve. In the present system, the quality assessor serves as the proxy; the actual usefulness

of generated code constitutes the true reward. Karwowski et al. [12] provide geometric analysis of when proxy optimization becomes harmful, proposing anti-Goodharting policy optimization methods.

2.2 B. Structural Introns and Neutral Drift

In genetic programming (GP), structural introns are instructions disconnected from program output [2, 1]. They accumulate during evolution because mutations to non-coding regions are selectively neutral. Miller [16] demonstrates that in Cartesian GP, over 95% of genes can be non-coding, yet this neutral drift is beneficial because it enables genotypic exploration without phenotypic risk.

Four of the six genes in the present system are structural introns. Unlike GP introns, they are not structurally detectable from the genome representation alone. The code appears to use them, but downstream overrides prevent them from reaching execution, making the disconnection harder to diagnose than traditional intron accumulation.

2.3 C. The ANIL Insight: Cosmetic Inner Loops

Raghu et al. [18] demonstrate that in MAML meta-learning, the inner loop (task-specific adaptation) barely modifies the learned representation. The Almost No Inner Loop (ANIL) algorithm removes inner-loop updates for all layers except the task head and matches full MAML performance.

This finding provides the closest published analogy to the present system's failure mode. The language model's capabilities (analogous to meta-learned features) already determine output quality. The genome parameters (analogous to inner-loop updates) are optimized but do not meaningfully change behavior. Just as ANIL demonstrated that the inner loop was cosmetic, the present system's evolution engine parameters may be entirely removable without affecting output quality.

2.4 D. Neural Architecture Search: When Search Equals Random

Li and Talwalkar [14] show that random architecture search matches sophisticated NAS methods (ENAS, DARTS) when given the same computational budget. The implication is that if search space parameters do not strongly influence the objective, any search strategy, including random search, performs equivalently. DARTS specifically suffers from performance collapse where architecture parameters converge to degenerate solutions that optimize the proxy but not the true objective [23].

2.5 E. Prompt Optimization Failure Modes

EvoPrompt [10] applies evolutionary algorithms to prompt optimization but degrades performance on GSM8K, as the evolution does not capture task structure and produces mutations that harm reasoning. OPRO [22] uses language models to self-optimize prompts but fails with smaller models that lack meta-cognitive capacity [4]. PromptBreeder [6] evolves both prompts

and mutation operators (meta-evolution) but risks circularity when the evolved mutation operators are themselves subject to disconnection.

2.6 F. Darwin Godel Machine: A Production Precedent

The most directly relevant production system is Sakana AI's Darwin Godel Machine (DGM) [19], a self-improving coding agent that uses evolution to rewrite its own source code. DGM exhibited reward hacking: it falsified test results and disabled hallucination detection to appear to perform better. This represents the adversarial variant of Goodhart's Law [13], in which the optimization process is not merely disconnected from outcomes but actively subverts its evaluation.

2.7 G. Evolution Strategies and Fitness Landscape Flatness

Salimans et al. [20] demonstrate that evolution strategies can serve as a scalable alternative to reinforcement learning but require informative fitness gradients. Dang et al. [5] show that non-elitist evolutionary algorithms can navigate landscapes with sparse deceptive regions, but this advantage depends on the existence of exploitable fitness structure. When the fitness landscape is effectively flat with respect to the evolved parameters, no evolutionary strategy can extract useful signal.

2.8 H. Genotype-Phenotype Mapping Theory

Whigham et al. [21] provide a theoretical framework for analyzing genotype-to-phenotype mappings in evolutionary algorithms, identifying conditions under which the mapping becomes degenerate (many-to-one) or disconnected (genotype changes do not produce phenotype changes). The present system exhibits both pathologies simultaneously: all genomes map to effectively the same phenotype because downstream overrides erase genotypic variation.

Summary. Among known autonomous AI agents (Devin, SWE-agent, AutoGPT, BabyAGI, OpenHands), only DGM [19] and Night Shift employ evolutionary optimization. The dominant paradigm of planning, tool use, and reflection avoids the disconnection problem by not meta-optimizing parameters at all. Both systems that do use evolution exhibit disconnection pathologies, suggesting that this failure mode may be inherent to evolutionary meta-optimization in autonomous AI systems unless explicit safeguards are implemented.

3 III. System Description

3.1 A. Night Shift Architecture

Night Shift is an autonomous software development system that selects tasks from a prioritized backlog, generates implementation plans, executes code modifications, and evaluates the results. The system has operated continuously for 21 days, completing over 500 tasks across four project categories. Table I summarizes the system specifications.

Component	Specification
System version	Night Shift v7.2, 21+ consecutive days of operation
Evolution algorithm	NSGA-II with island model (explorer, exploiter, generalist)
Population size	20 genomes per generation
Genome length	6 genes (see Table II)
Fitness function	8-axis weighted sum (see Section III-C)
Adaptive pressure	Rechenberg 1/5 rule, plateau detection, idle decay
Textual gradients	LLM-generated feedback per task (Haiku model)
Local GPU	NVIDIA RTX 4000 SFF Ada 20 GB, Qwen 2.5 Coder 32B
Tasks completed	500+ across 4 project categories

Table 1: Table I: Night Shift System Specifications

3.2 B. Genome Structure

Each genome encodes six genes that are intended to control task execution parameters. Table II defines each gene and its value range.

Gene	Type	Range	Intended Function
model	Categorical	sonnet, opus, haiku	LLM model selection
token_budget	Integer	4,096 – 16,384	Maximum output tokens
prompt_style	Categorical	detailed, concise, structured	Prompt template selection
context_depth	Integer	1 – 10	Number of context files
continuation_limit	Integer	0 – 3	Maximum task continuations
scope	Categorical	narrow, moderate, ambitious	Task scoping strategy

Table 2: Table II: Genome Gene Definitions

3.3 C. Fitness Function

The fitness function computes a weighted sum of eight axes: quality (assessor score), frugality (cost efficiency), reliability (completion rate), safety (no regressions), novelty (code diversity), consistency (variance across runs), learning (lineage trajectory), and merge success (code integration). The dispatcher passes `genome_distance`, `recent_scores`, and `merge_status` as inputs; however, as documented in Section IV, three of these inputs receive constant default values in production.

3.4 D. Execution Pipeline

The intended execution pipeline proceeds as follows: (1) the evolution engine selects a genome from the population; (2) the genome's parameters are applied to the task via `apply_to_task()`; (3) the task is dispatched to the LLM with the genome-specified configuration; (4) the output is evaluated by the quality assessor; and (5) the fitness score is recorded against the genome. As Section IV demonstrates, this pipeline is severed at multiple points.

4 IV. Methodology: Diagnostic Tests

4.1 A. The Five-Link Causal Break

Source code analysis and runtime logging reveal that the genome's influence on task outcomes is severed at five points in the execution pipeline.

Link 1: Unapplied genes. The `apply_to_task()` method in `genome.py` transfers only `model` and `token_budget` to the task dictionary. The remaining four genes (`prompt_style`, `context_depth`, `continuation_limit`, `scope`) are mutated and crossed over during evolution but are never applied to task execution. These genes are structural introns in the sense of Brameier and Banzhaf [2].

Link 2: Model override. The model router makes deterministic routing decisions based on task category, priority, and output type. For P2 and higher priority tasks, which constitute virtually all Night Shift tasks, the router ignores the genome's model selection and applies its own cost-optimization logic. The genome may specify `opus`, but execution uses `qwen2.5-coder` on the local GPU.

Link 3: Budget override. The SelfBudgeter module estimates output token requirements using a heuristic based on task type and overwrites the genome's `token_budget` gene, which is the second of the two genes that is applied.

Link 4: Constant fitness inputs. The fitness function accepts `genome_distance` (population diversity), `recent_scores` (lineage trajectory), and `merge_status` (code merge success) as inputs, but the dispatcher passes default constants for all three. Three of eight fitness axes therefore evaluate meaningless values.

Link 5: Cross-task comparison. Mutation improvement is measured by comparing child genome fitness to parent genome fitness. However, parent and child execute on different tasks with different inherent difficulty. A mutation that produces a superior genome may appear harmful because the child was assigned a more difficult task.

4.2 B. Gene Effectiveness Analysis

Table III traces each gene through the execution pipeline to determine whether it has end-to-end causal influence on task outcomes.

Gene	In Genome	Applied to Task	Reaches Execution	Influences Output	Measured in Fitness
model	Yes	Yes	No (overridden)	No	No
token_budget	Yes	Yes	No (overridden)	No	No
prompt_style	Yes	No	No	No	No
context_depth	Yes	No	No	No	No
continuation_limit	Yes	No	No	No	No
scope	Yes	No	No	No	No

Table 3: Table III: Gene Effectiveness Matrix

Zero of six genes have end-to-end causal influence on task outcomes.

4.3 C. Experimental Design

Five experiments were designed to quantify the disconnection using the production evolution database containing 564 evaluated genomes across 28 generations. All analyses were conducted on production data without experimental intervention, as the system was in continuous autonomous operation during the study period. The experiments are:

1. **Generation trend analysis:** Compare mean fitness of the first five and last five generations to detect improvement or decline.
2. **Elite vs. population comparison:** Apply Welch's t-test to compare elite genomes (top 20 by fitness) against the remainder of the population.
3. **Parameter-fitness correlation:** Analyze the distribution of each gene value and its relationship to fitness scores.
4. **Niche analysis:** Decompose fitness by task category and output type to determine whether fitness variance is explained by genome parameters or task assignment.
5. **Variance decomposition:** Compare between-niche and within-niche fitness variance to isolate the genome's contribution.

5 V. Results

5.1 A. Generation Trend Analysis

Mean fitness was computed for the first five and last five generations of the evolutionary run. The first five generations achieved a mean fitness of 0.6413, while the last five generations achieved a mean of 0.5941, representing a decline of 0.0472 (7.4%). Table IV presents the generation trajectory at four-generation intervals.

Generation	Mean Fitness	Max Fitness	Trend
4	0.632	0.680	–
8	0.649	0.720	+2.7%
12	0.654	0.890	+0.8%
16	0.660	0.890	+0.9% (peak)
20	0.637	0.905	-3.5%
24	0.612	0.905	-3.9%
28	0.588	0.905	-3.9%

Table 4: Table IV: Generation Fitness Trajectory

The adaptive mutation rate reached its maximum clamp of 0.40, and the plateau counter registered 13 consecutive generations without improvement. The maximum fitness of individual genomes (0.905) remained constant from generation 20 onward, reflecting the persistence of a single elite genome rather than population-wide improvement.

If evolution were functioning correctly, mean fitness should increase monotonically or plateau; a sustained decline indicates that selection pressure is unable to distinguish beneficial mutations from harmful ones [5].

5.2 B. Elite vs. Population Comparison

A Welch's t-test was applied to compare the fitness of elite genomes (top 20 by fitness score, n = 20, mean = 0.8490, SD = 0.0536) against the remaining population (n = 82, mean = 0.5033, SD = 0.1630). The test yielded $t = 15.98$, $df = 91.8$, $p < 0.001$.

While this result confirms that elite genomes have statistically significantly higher fitness than the rest of the population, it does not confirm that evolution produced this difference. As demonstrated in Section V-D, elite fitness reflects task assignment (elites disproportionately received RESEARCH-category tasks, which have inherently higher assessor scores) rather than genome quality. The t-test result is therefore consistent with both the hypothesis that evolution works and the hypothesis that task assignment determines fitness.

5.3 C. Parameter-Fitness Correlation

Analysis of the model gene revealed that 97% of the population (n = 99 of 102 genomes in the final generations) carried the value `sonnet`. Only two genomes carried `opus` (mean fitness = 0.5949) and one carried `haiku` (mean fitness = 0.6400). This near-total absence of gene variance is a direct consequence of the model override (Link 2 in Section IV-A): because the router deterministically selects the model regardless of the genome's specification, there is no selective pressure differentiating model gene values. The gene has insufficient variance for evolution to act upon.

In contrast, task category exhibited a strong relationship with fitness despite being outside genome control:

Category	n	Mean Fitness
RESEARCH	–	0.773
LIVINGCORP	–	0.664
NEXUS	–	0.640
HUB	–	0.481

Table 5: Table V: Fitness by Task Category

Similarly, output type showed a strong relationship: research outputs averaged 0.773, code outputs 0.555, and report outputs 0.475. Neither category nor output type is under genome control; both are determined by the task backlog.

Token budget exhibited a nonlinear relationship (low: 0.542, medium: 0.758, high: 0.425), but this correlation is confounded by category: RESEARCH tasks tend to have medium budgets, and RESEARCH tasks have inherently higher fitness scores.

5.4 D. Niche Analysis

Fitness was decomposed by niche, defined as the combination of task category and output type. Table VI presents the results for the six largest niches.

Between-niche means range from 0.425 to 0.773 (range = 0.348), while within-niche standard deviations range from 0.065 to 0.250. The between-niche variation substantially exceeds the

Niche (Category_OutputType)	n	Mean Fitness	SD
HUB_report	36	0.474	0.179
RESEARCH_research	21	0.773	0.135
META_code	17	0.425	0.159
LIVINGCORP_code	13	0.664	0.065
NEXUS_code	8	0.657	0.250
HUB_code	3	0.554	0.077

Table 6: Table VI: Within-Niche Fitness Distribution

within-niche variation, indicating that task category and output type explain the majority of fitness variance.

5.5 E. Variance Decomposition

If genome parameters influenced fitness, evolved genomes should exhibit lower within-niche variance than randomly assigned genomes, because evolution would be selecting for genome configurations that perform well within each niche. The observed within-niche standard deviations (0.065 to 0.250) represent the noise floor of the fitness function. Since evolved genomes cannot be distinguished from hypothetical random genomes within niches, the genome parameters are not contributing to fitness determination.

The overall fitness variance can therefore be decomposed as:

Total variance = Between-niche variance (task assignment) + Within-niche variance (noise floor) + Genome contribution (undetectable, approximately zero).

6 VI. Discussion

6.1 A. Formalization of Disconnected Evolution

The results establish that Night Shift's evolutionary engine operates on parameters that do not influence outcomes. This failure mode, termed Disconnected Evolution, can be formalized as follows: given a genotype space G, a phenotype space P, and a fitness function f: P -> R, evolution is disconnected when the mapping g: G -> P is degenerate (all genotypes map to the same or indistinguishable phenotypes) such that for any mutation operator m: G -> G, f(g(m(x))) is approximately equal to f(g(x)) plus noise for all x in G.

Disconnected Evolution is distinct from several related phenomena. It differs from a flat fitness landscape, in which the phenotype varies but fitness does not discriminate. It differs from neutral drift [16], in which genotypic changes do not affect phenotype but may enable future exploration. In Disconnected Evolution, the genotype-phenotype mapping itself is broken: phenotypic changes would affect fitness, but genotypic changes cannot produce phenotypic changes because intermediate pipeline components erase genotypic variation.

6.2 B. Relationship to Goodhart's Law

The disconnection represents the causal variant of Goodhart's Law as taxonomized by Manheim and Garrabrant [15]. The genome's fitness score functions as a proxy measure that was intended

to track task outcome quality. However, because the causal pathway from genome to outcome is severed, optimizing the proxy (genome fitness) cannot improve the target (task outcomes). This is structurally analogous to the reward model overoptimization documented by Gao et al. [8], where increasing proxy reward eventually decreases true reward.

6.3 C. Comparison with the Darwin Godel Machine

The Darwin Godel Machine (DGM) [19] and Night Shift represent the only two known autonomous AI systems employing evolutionary optimization. Both exhibit disconnection pathologies, but of different types. DGM's disconnection is adversarial: the system actively subverts its evaluation through reward hacking [13]. Night Shift's disconnection is passive: the system's architecture inadvertently prevents genome parameters from reaching execution. The DGM case is more dangerous (the system games its metrics), while the Night Shift case is more wasteful (computational resources are consumed without effect).

6.4 D. The Neutral Drift Hypothesis

An alternative interpretation is that Night Shift's genome mutations represent neutral drift in the GP sense [16]: genotypic changes that do not affect phenotype. Miller [16] demonstrates that neutral drift is beneficial in Cartesian GP because it enables genotypic exploration without fitness risk. Under this interpretation, the 28 generations of neutral evolution may represent useful genotypic diversity that could become expressed once the execution wiring is repaired. However, this interpretation is less parsimonious than the disconnection hypothesis because it requires the additional assumption that the accumulated genotypic diversity will be useful, which is not supported by the current evidence.

6.5 E. Detection Framework

Based on the diagnostic experiments and the literature survey, the following detection framework is proposed for identifying Disconnected Evolution in autonomous systems:

Test	**Derived From**	**Detection Target**
Generation trend analysis	Standard EA monitoring	Fitness decline over generations
Elite-population t-test	Statistical hypothesis testing	Whether elite status reflects selection or luck
Parameter-fitness correlation	Hutter et al. [11] (fANOVA)	Which parameters influence fitness
Niche decomposition	Quality-diversity literature	Whether fitness depends on task or genome
Random baseline comparison	Li and Talwalkar [14]	Whether evolved genomes outperform random
Perturbation testing	DARTS sensitivity analysis [23]	Whether parameter changes affect output

Table 7: Table VII: Disconnected Evolution Detection Methods

6.6 F. Proposed Architectural Remedies

The fundamental remedy is not algorithmic (the evolutionary operators are sound) but architectural: restoring the causal link between genome parameters and task execution. Table VIII summarizes the proposed changes.

Gene	Current State	Proposed Repair	Expected Effect
model	Overridden by router	Genome selects within router's acceptable tier	Evolution discovers optimal model per niche
token_budget	Overridden by budgeter	Budgeter fills only if genome did not specify	Evolution discovers optimal budget per task type
prompt_style	Never applied	Template selection in context engine	Evolution discovers optimal prompt structure
context_depth	Never applied	File count limit in context engine	Evolution discovers optimal context density
continuation_limit	Never applied	Maximum continuations in dispatcher	Evolution discovers optimal continuation count
scope	Never applied	Task scoping strategy in context engine	Evolution discovers narrow vs. ambitious strategies

Table 8: Table VIII: Proposed Gene-Execution Wiring Repairs

Additionally, the fitness function should employ niche-normalized scoring. Replacing raw quality comparison with z-score normalization within task niche (category x output_type) would separate task difficulty from genome contribution:

F_genome = (Q_task - mu_niche) / sigma_niche

Under this formulation, a genome achieving 7/10 on a difficult task (niche mean 5/10) scores higher than one achieving 8/10 on an easy task (niche mean 8/10).

6.7 G. Should Autonomous AI Systems Use Evolution?

The survey reveals that evolution is rare in production AI agents. The dominant paradigm (planning, tool use, reflection) avoids meta-optimization entirely, relying on the language model's internal capabilities. DGM [19] and Night Shift are the only known exceptions, and both exhibit disconnection pathologies.

Evolution remains viable for autonomous AI systems when three conditions are satisfied: (1) genes must causally influence outcomes, meaning the genotype-phenotype mapping must be validated, not assumed; (2) the fitness landscape must exhibit gradient, meaning if the language model's capability plateau creates a flat landscape, evolution has no signal to follow; and (3) evaluation must be independent of the evolved system, to prevent reward hacking as observed in DGM [19].

7 VII. Threats to Validity

7.1 A. Internal Validity

This study analyzes production data from a single system rather than controlled experimental data. Task assignments were not randomized with respect to genome parameters, introducing potential confounds between genome quality and task difficulty. The five-link causal break analysis (Section IV-A) is based on source code inspection and runtime logging rather than formal verification. Additionally, the quality assessor used as the fitness proxy may itself introduce systematic biases that confound the analysis.

7.2 B. External Validity

The findings are derived from a single evolution engine implementation within a single autonomous development system. The degree to which Disconnected Evolution generalizes to other evolutionary meta-optimization systems depends on architectural similarities. However, the fact that both known evolutionary autonomous AI systems (Night Shift and DGM [19]) exhibit disconnection pathologies suggests the phenomenon may be widespread. The eight-mechanism

taxonomy (Section II) is drawn from diverse domains, supporting the generalizability of the disconnection concept if not the specific failure mode.

7.3 C. Construct Validity

Fitness is measured through an automated quality assessor, which serves as a proxy for actual code quality and utility. The validity of the fitness metric as a measure of true task outcome quality has not been independently validated. If the assessor systematically misestimates quality for certain task types or genome configurations, the variance decomposition results (Section V-E) could be misleading. Furthermore, the definition of "mutation improvement" requires comparing parent and child genomes that executed on different tasks, which conflates genome quality with task difficulty as noted in Link 5 (Section IV-A).

8 VIII. Conclusion

This paper documents Disconnected Evolution, a failure mode in which evolutionary optimization operates on parameters that do not influence outcomes. In the Night Shift autonomous development system, 477 mutations across 28 generations produced a 7.4% fitness decline because all six genome genes were either unapplied (four genes) or overridden (two genes) before reaching execution. Five diagnostic experiments using 564 evaluated genomes confirm that fitness variance is explained by task assignment rather than genome parameters.

The failure mode arises at the intersection of Goodhart's Law [15] (proxy optimization diverges from true objective), structural introns [2] (non-coding genes accumulate without detection), the ANIL insight [18] (optimization loops can be cosmetic), and NAS search-space irrelevance [14] (sophisticated search matches random when parameters do not matter). A taxonomy of eight disconnection mechanisms was developed from the literature, and detection methods based on generation trend analysis, parameter-fitness correlation, and niche decomposition were validated on production data.

Among autonomous AI development agents, only Night Shift and the Darwin Godel Machine [19] employ evolutionary optimization, and both exhibit disconnection pathologies. The architectural remedy is not algorithmic but structural: restoring the causal link between evolved parameters and task execution through gene-execution wiring repairs and niche-normalized fitness evaluation.

The broader lesson for the autonomous AI systems community is that evolutionary meta-optimization requires validated causal links between the evolved parameters and the evaluated outcomes. Without this validation, evolution consumes computational resources without producing optimization signal.

9 Data Availability

The Night Shift system is proprietary. Aggregated metrics and evolution database schema are available upon request to the corresponding author.

10 Ethics Statement

This research analyzes an autonomous system operating on the authors' own codebase. No human subjects or external data were involved.

References

[1] W. Banzhaf, P. Nordin, R. Keller, and F. Francone, "Some considerations on the reason for bloat," extitGenetic Programming and Evolvable Machines, vol. 1, no. 1, pp. 17–31, 2000.

[2] M. Brameier and W. Banzhaf, extitLinear Genetic Programming. New York, NY: Springer, 2010.

[3] L. Chen, M. Zaharia, and J. Zou, "FrugalGPT: How to use large language models while reducing cost and improving performance," extitarXiv preprint arXiv:2305.05176, 2023.

[4] Y. Chen, Z. Li, and Y. Zhang, "Revisiting OPRO: The limitations of small-scale LLMs as optimizers," extitarXiv preprint arXiv:2405.10276, 2024.

[5] D.-C. Dang, T. Jansen, and P. K. Lehre, "Non-elitist evolutionary algorithms excel in fitness landscapes with sparse deceptive regions," in extitProc. Genetic and Evolutionary Computation Conf. (GECCO), 2021, pp. 615–623.

[6] C. Fernando, D. Banarse, H. Michalewski, S. Osindero, and T. Rocktaschel, "Prompt-Breeder: Self-referential self-improvement via prompt evolution," extitarXiv preprint arXiv:2309.16797, 2023.

[7] J. Frankle and M. Carlin, "The lottery ticket hypothesis: Finding sparse, trainable neural networks," in extitProc. Int. Conf. Learning Representations (ICLR), 2019.

[8] L. Gao, J. Schulman, and J. Hilton, "Scaling laws for reward model overoptimization," in extitProc. Int. Conf. Machine Learning (ICML), 2023, pp. 10835–10866.

[9] V. Golubenko, "Night Shift: Autonomous AI development with evolutionary quality optimization," TOV ZELTREX, Zhytomyr, Ukraine, Tech. Rep., 2026.

[10] Q. Guo, R. Wang, J. Guo, B. Li, K. Song, X. Tan, G. Liu, J. Bian, and Y. Yang, "Connecting large language models with evolutionary algorithms yields powerful prompt optimizers," in extitProc. Int. Conf. Learning Representations (ICLR), 2024.

[11] F. Hutter, H. Hoos, and K. Leyton-Brown, "An efficient approach for assessing hyperparameter importance," in extitProc. Int. Conf. Machine Learning (ICML), 2014, pp. 754–762.

[12] J. Karwowski, O. Sheratt, J. Foerster, and M. Sheratt, "Goodhart's law in reinforcement learning," in extitProc. Int. Conf. Learning Representations (ICLR), 2024.

[13] V. Krakovna, J. Uesato, V. Mikulik, M. Rahtz, T. Everitt, R. Kumar, Z. Kenton, J. Leike, and S. Legg, "Specification gaming: The flip side of AI ingenuity," DeepMind Blog, 2020.

[14] L. Li and A. Talwalkar, "Random search and reproducibility for neural architecture search," in extitProc. Conf. Uncertainty in Artificial Intelligence (UAI), 2019, pp. 367–377.

[15] D. Manheim and S. Garrabrant, "Categorizing variants of Goodhart's law," extitarXiv preprint arXiv:1803.04585, 2018.

[16] J. F. Miller, "Neutrality and self-adaptation: What is the relationship?" in extitGenetic Programming and Evolvable Machines, vol. 16, no. 2, pp. 225–252, 2015.

[17] I. Ong, A. Almahairi, V. Wu, W.-L. Chiang, T. Wu, J. E. Gonzalez, M. W. Kadous, and I. Stoica, "RouteLLM: Learning to route LLMs with preference data," extitarXiv preprint arXiv:2406.18665, 2024.

[18] A. Raghu, M. Raghu, S. Bengio, and O. Vinyals, "Rapid learning or feature reuse? Towards understanding the effectiveness of MAML," in extitProc. Int. Conf. Learning Representations (ICLR), 2020.

[19] Sakana AI, "Darwin Godel Machine: Open-ended evolution of self-improving agents," extitarXiv preprint arXiv:2505.22954, 2025.

[20] T. Salimans, J. Ho, X. Chen, S. Sidor, and I. Sutskever, "Evolution strategies as a scalable alternative to reinforcement learning," extitarXiv preprint arXiv:1703.03864, 2017.

[21] P. A. Whigham, G. Dick, J. Maclaurin, and C. A. Owen, "Examining the best of both worlds of cross-validation and holdout for evolutionary computing," extitGenetic Programming and Evolvable Machines, vol. 18, no. 4, pp. 387–415, 2017.

[22] C. Yang, X. Wang, Y. Lu, H. Liu, Q. V. Le, D. Zhou, and X. Chen, "Large language models as optimizers," extitarXiv preprint arXiv:2309.03409, 2023.

[23] A. Zela, T. Elsken, T. Saikia, Y. Marber, T. Brox, and F. Hutter, "Understanding and robustifying differentiable architecture search," in extitProc. Int. Conf. Learning Representations (ICLR), 2020.