

A corrective constraint increases multi-target attainment under finite adaptive search

Matthew E. Brashears¹, Eric Gladstone^{2*}, Jose Ferrer³

¹ Department of Sociology, McCausland College of Arts and Sciences, University of South Carolina, Columbia, SC, USA

² Independent Researcher, Traverse City, MI, USA

³ School of Computer Science, Georgia Institute of Technology, Saint George, UT, USA

*Corresponding author: Eric Gladstone, eric.c.gladstone@gmail.com

ORCID iDs: Matthew E. Brashears, <https://orcid.org/0000-0003-1029-4682>. Eric Gladstone, <https://orcid.org/0000-0003-4393-050X>.

Keywords: error correction, constrained variation, adaptive search, fitness landscapes, complex adaptive systems

Abstract

Correction can accelerate adaptive search without knowing which targets are being sought, provided its admissible set preserves components those targets require. We test this in a textual replicator model with targets, target-specific retention, and a dictionary-based corrector blind to target identity. Across 320 runs per condition, correction increased the mean number of exact targets reached within 500 generations from 0.38 to 0.90. Extended runs showed that much of this advantage reflected faster attainment rather than an established increase in long-run coverage. Point estimates favored correction in all nine sampled landscapes, with smaller advantages at wider peak spacing. Removing words required by the targets eliminated the benefit and made correction harmful. Process analyses located differences before and after entry into the near-target region without isolating a causal stage. Within an architecture that preserves competing trajectories, a corrective constraint can increase finite-horizon multi-target attainment when the admissible set preserves required solution components.

Introduction

Replicating systems must preserve information while remaining able to change (1, 2). Too little variation limits adaptation, while too much erodes structure. That tension sharpens when search is bounded. Reaching a new solution may require producing many failures, so a finite budget of generations can end before many adaptive outcomes in the landscape are realized (3, 4, 5). When mutation repeatedly disrupts partial structure, the problem is not simply to generate variation, but to generate it without continually losing intermediate states that retain useful structure (6, 7, 8, 9).

Correction offers one response to that problem. Rather than simply lowering the mutation rate, it redistributes variation toward an admissible set (10, 11). A spell checker provides a simple example. A sentence belongs to the large space of possible character strings, but a spell checker maps malformed words toward the much smaller set of strings composed of valid words. This restriction can make plausible forms more likely without identifying the intended sentence.

Correction can therefore improve the chance of reaching a target without requiring the system to know which target it is seeking.

Related literatures already show that restricting or structuring variation need not impede search. In evolutionary computation, repair operators map infeasible candidates back toward admissible regions of a search space (12). In evolutionary biology, developmental bias and facilitated variation emphasize that developmental systems generate some viable phenotypes more readily than others, while work on robustness and evolvability shows that preserving function under perturbation can alter which novel phenotypes become accessible to evolving populations (9, 11, 13). Multimodal optimization addresses a complementary population-level problem. Niching methods preserve competing subpopulations so that a search can locate several optima rather than collapse onto one (14). These traditions make the broad proposition that structured variation or protected alternatives can facilitate search unsurprising.

The narrower question is whether an active local correction rule can increase exact attainment across multiple distinct targets within a fixed search horizon, and how any advantage varies with search time, target arrangement, and whether the admissible set preserves components required by the targets. Earlier unpublished work by this author team, using the original Java implementation, examined correction under global selection. That design does not provide the present test because it retains only the fittest replicators across all targets, causing the population to converge on whichever target is easiest to reach. We therefore retain the earlier model's mutation, correction, and fitness framework while replacing global selection with target-specific retention, allowing several target-associated lineages to persist at once. We ask whether correction increases the number of targets reached within a finite horizon, whether the advantage remains positive across sampled target arrangements, and where along the fitness trajectory the correction difference appears.

We study this question in a population of textual replicators evolving toward target sentences that define peaks in a fitness landscape (15, 16). The substrate is deliberately transparent. Variation, correction, fitness evaluation, and selection are explicit operations, and the primary attainment outcome is an exact match rather than a threshold on a continuous score. Two features of the design matter for interpretation. Target-specific retention preserves several competing trajectories by construction, so it defines the experimental regime in which multi-target attainment can occur. The corrector itself is target-blind during a run, although its dictionary contains the vocabulary used in the targets.

Correction increases the number of distinct targets reached within the standard horizon. Extended runs indicate that much of this advantage reflects faster attainment, while the average advantage becomes smaller across more widely spaced sampled landscapes. The effect also depends on the correction dictionary preserving words required by the targets. Removing those words eliminates the advantage and makes correction harmful. Process differences appear during both approach to the near-target region and completion, although the analyses do not isolate a causal stage. Finally, reducing mutation without correction does not reproduce the correction advantage within the sampled range of mutation multipliers.

The Model

We model a population of textual replicators, each represented by an English sentence, evolving toward target sentences that define peaks in a fitness landscape. The model separates variation, correction, fitness evaluation, and selection so that each operation can be specified explicitly (Fig. 1).

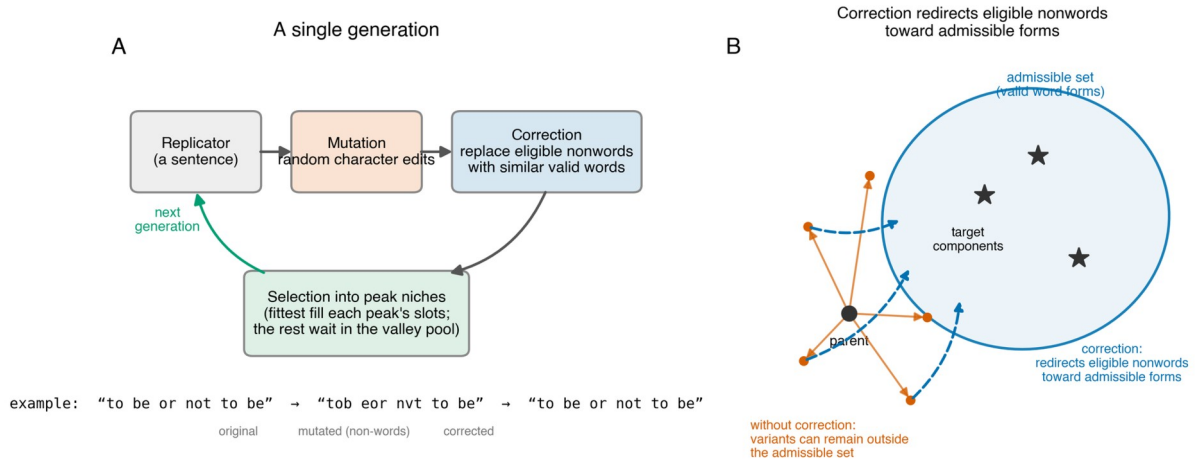


Fig. 1. Model structure and corrective constraint. (A) Each generation, a replicator produces mutated copies. In the correction condition, each copy passes through a dictionary that replaces eligible nonwords with sufficiently similar valid words. Selection retains the fittest replicators within target-specific niches, while the remainder enter the shared valley remnant. (B) Correction redirects eligible malformed variants toward an admissible set of valid word forms containing the target components. Without correction, mutated variants are not redirected toward that lexical set and can remain as nonword states.

Alt text: Two-panel schematic of the model. Panel A shows a replicator producing mutated offspring, optional dictionary correction, and selection into target-specific peak niches or the shared valley remnant before the next generation. Panel B shows correction redirecting eligible malformed variants toward valid word forms containing target components, while uncorrected mutation can leave variants outside that lexical set, including nonwords.

Variation. Each replicator produces ten mutated copies per generation. Mutation acts on characters and spaces through fixed base rates for substitution, deletion, and insertion. A single mutation-rate multiplier scales all of these rates. It therefore controls copying noise while landscape geometry remains fixed by the targets and fitness function. As copying noise rises, partial solutions are more likely to be disrupted before selection (6).

Correction. In the correction condition, each mutated copy passes through a dictionary. A word absent from the dictionary is replaced by a randomly selected valid word drawn from candidates with sufficiently similar spelling. When no candidate is sufficiently close, the original nonword is retained. The corrector does not reference target identities or target sentences during a run, although its dictionary contains the vocabulary used in the targets. It therefore biases mutation outcomes toward valid words without steering toward a particular target. The consequences of this codebook structure are examined below.

Fitness and peaks. Fitness is the similarity between a replicator and its nearest target, calculated from character-level edit similarity (17) and set-based word overlap. The environment contains 100 candidate targets, of which a selected subset defines the active peaks in each run. A peak is colonized when a replicator assigned to it exactly matches the target. The targets range from 53 to 109 characters and have a median pairwise edit distance of 60. This string-evolution design follows the cumulative-selection tradition (15, 16). Because correction acts on lexical form while fitness includes both word overlap and character-level edit similarity, correction and evaluation

are structurally aligned in this representation. The representation is held constant between conditions, and this structural compatibility is not independently manipulated.

Multi-peak selection. The earlier model used global selection, which retained only the fittest replicators across all targets and led the population to converge on the easiest target. Here, each replicator is instead assigned to its nearest peak. Each peak contains a fixed number of slots that retain distinct genotypes. The fittest replicators assigned to a peak fill those slots, while the remainder enter a shared pool of between-peak replicators called the valley remnant. This architecture allows lineages associated with several targets to persist simultaneously (18, 19, 20). For each colonization, we record whether the successful lineage arrived from the valley remnant, matured in place within a peak, or transferred from another peak. Full parameter values and procedures appear in Methods.

Results

Corrective constraint increases multi-target attainment under finite search

We first tested whether local correction changes multi-target attainment under target-specific retention, using a close cluster of five target peaks and a 500-generation search horizon (Fig. 2). Across 320 runs per condition, correcting populations reached a mean of 0.90 peaks (SE = 0.04, 95% CI 0.82 to 0.98), compared with 0.38 without correction (SE = 0.03, 95% CI 0.32 to 0.44). The difference was 0.52 peaks per run (95% CI 0.43 to 0.62), a 137 percent increase over the no-correction mean. Correction also shifted the distribution of outcomes: 71 percent of correcting runs reached at least one peak, compared with 36 percent without correction, while 18 percent reached at least two peaks, compared with 2 percent.

Point estimates favored correction at all four sampled mutation-rate multipliers. The advantage was largest at the two intermediate values, 0.5 and 1.0, and smaller at 0.25 and 1.5. With only four sampled multipliers, this pattern does not identify an optimum, but it shows that the finite-horizon advantage is not confined to a single mutation level.

The result also persists under less stringent attainment criteria. In the separate 20-run close-design sensitivity basis, defining attainment as reaching 0.90 or 0.95 of maximum fitness produced correction differences of 1.70 and 1.00 peaks, compared with 0.65 under exact matching. These are nested outcomes from the same runs rather than independent replications, but they show that the positive difference is not an artifact of requiring exact completion.

Reducing mutation without correction did not reproduce the correction result within the tested range. In this sensitivity analysis, correction at multiplier 1.0 reached 1.11 peaks, whereas the highest observed no-correction mean was 0.64 peaks at multiplier 0.5 (Supplementary Information S12). Because that comparator was selected after inspecting all four no-correction conditions, we do not attach a confidence interval to the across-multiplier comparison. The bounded conclusion is that lower mutation alone did not match the attainment observed under correction among the four sampled multipliers.

As an implementation check, the current model also reproduces the single-target acceleration pattern first obtained with the original implementation under global selection (Supplementary Information S16, Fig. S4).

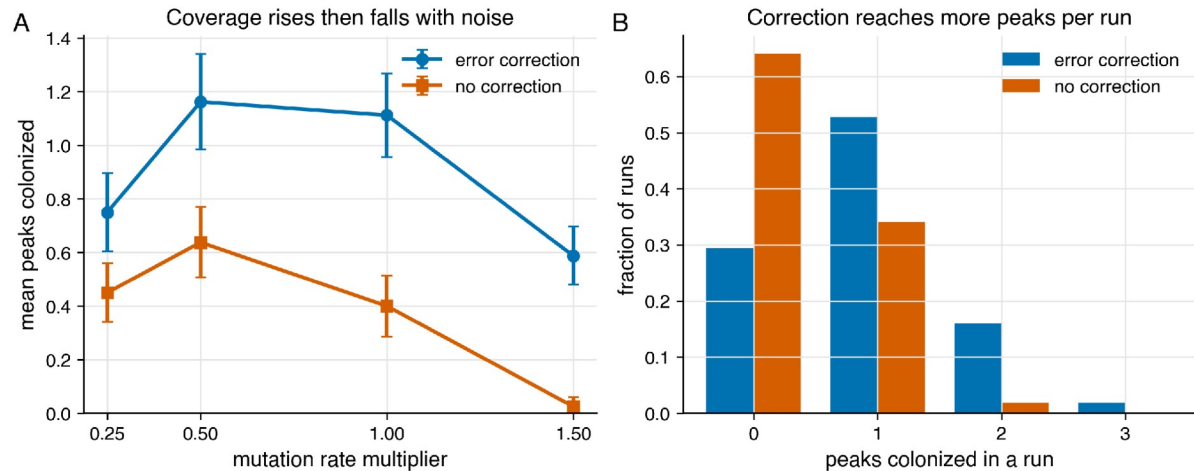


Fig. 2. Correction increases multi-target attainment under finite search. (A) Mean number of peaks reached across the four sampled mutation-rate multipliers, with and without correction, using the pooled 320 runs per condition. Error bars show 95% confidence intervals. (B) Distribution of the number of peaks reached per run. Correcting populations reach more peaks and are less likely to end without reaching any peak.

Alt text: Two-panel comparison of multi-target attainment. Panel A shows higher mean attainment under correction at all four sampled mutation-rate multipliers, with the largest differences at the two intermediate values. Panel B shows that correction shifts runs away from zero attainment and toward one or more peaks.

The advantage reflects faster attainment and weakens across wider sampled landscapes

The 500-generation endpoint cannot distinguish faster attainment within a finite budget from a higher long-run coverage ceiling. We therefore extended the close-peak design to 1,500 generations (Fig. 3A). Correction retained a lead in reaching at least one or at least two peaks throughout the observed interval. By generation 1,500, correcting populations had reached a mean of 2.92 peaks, compared with 2.20 without correction (Fig. S1). The difference grew to about one peak near generation 1,000 and then narrowed to about 0.7 by generation 1,500 as the no-correction condition partly caught up. Much of the finite-horizon advantage therefore reflects faster attainment. Both conditions remained below the five-peak ceiling, so the observed runs do not establish their eventual coverage or relative asymptotes.

We next asked whether the result depended on a single target arrangement by sampling three independent landscapes at each of three mean distances between peaks. Point estimates favored correction in all nine landscapes (Fig. 3B), although several landscape-specific intervals included zero. The average difference declined from about 0.6 peaks at the closest spacing to about 0.1 at the widest. Because the sampled target sets differ in properties other than mean peak distance, this pattern does not isolate spacing as the cause of the decline. It shows instead that the correction advantage was smaller in the more widely spaced landscapes sampled here. Overall attainment also declined with spacing, reducing the opportunity for either condition to reach additional targets within the fixed horizon.

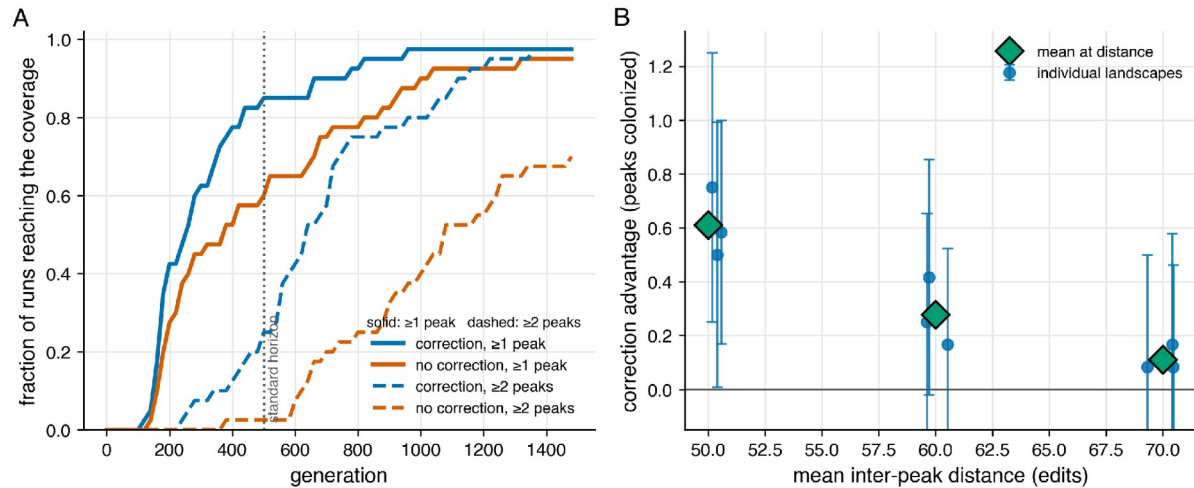


Fig. 3. Timing and landscape dependence of the correction advantage. (A) Fraction of runs reaching at least one peak, shown by solid lines, or at least two peaks, shown by dashed lines, as a function of generation. The close-peak design was run for 1,500 generations, with 80 runs pooled across mutation-rate multipliers 0.5 and 1.0. The dotted vertical line marks the standard 500-generation horizon. (B) Difference between correction and no correction in the number of peaks reached across nine independently sampled landscapes, with three landscapes at each of three mean distances between peaks and 12 runs per condition per landscape. Small circles show individual landscapes with slight horizontal displacement, and diamonds show the mean difference at each distance. Point estimates favor correction in all nine landscapes, while the average difference declines with spacing. Several landscape-specific intervals include zero.

Alt text: Two-panel analysis of attainment over time and across landscapes. Panel A shows correcting populations reaching at least one and at least two peaks earlier than populations without correction over 1,500 generations. Panel B shows positive estimated correction differences across nine sampled landscapes, with smaller average differences at wider mean peak spacing.

Correction depends on codebook support for required target components

The correction advantage depended on the dictionary preserving components required by the target solutions. In the standard 20-run close-design basis used for this intervention, removing the words used in the five active targets reduced mean attainment under correction from 1.20 peaks to zero, compared with 0.55 without correction (Fig. 4B). Reduced-dictionary correction also capped mean best fitness at about 0.76, compared with about 0.98 without correction (Fig. 4C), because target words generated during search were treated as errors and replaced with other dictionary entries. Codebook support for required target components is therefore a boundary condition of the correction advantage under the present operator. When those components are excluded, correction becomes harmful. Because the intervention both removes support for target words and actively rejects them when they appear, it does not estimate a gradual effect of codebook compatibility.

We also varied the number of genotype slots retained per peak (Fig. 4A). Attainment increased with slot count in both conditions, while correction retained a positive advantage at every tested capacity. Increasing the number of slots also increases the retained population and the number of offspring produced each generation (Supplementary Information S6), so the sweep changes

protected retention and total search effort together. It therefore does not isolate niche capacity as the cause of changes in attainment. The correction and no-correction conditions remain matched within each capacity, however, so the correction contrast remains interpretable within each tested capacity.

Finally, varying the dictionary-candidate similarity threshold produced the same 0.65-peak correction difference at thresholds of 0.70 and 0.80 and a smaller 0.30 difference at 0.90 (Supplementary Information S12). Across these three sampled thresholds, the estimated correction difference remained positive, but the analysis does not identify an optimal threshold.

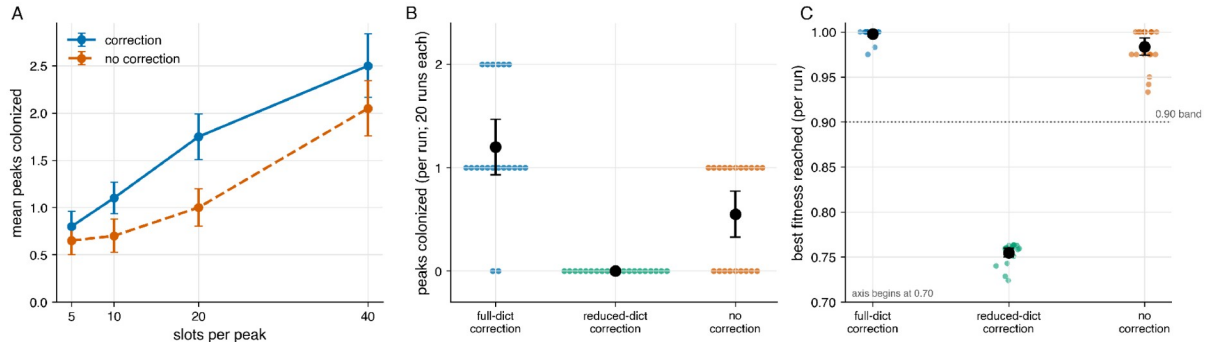


Fig. 4. Codebook support and genotype-slot sensitivity. (A) Mean number of peaks reached as a function of genotype slots per peak, with and without correction, with 95% confidence intervals and about 40 runs per condition at each capacity. (B) Peaks reached per run under full-dictionary correction, correction after removing the active target words from the dictionary, and no correction. Each point represents one of 20 runs per condition. Black markers show the mean and 95% interval. (C) Best fitness reached per run in the same three conditions. The dotted line marks the 0.90 near-target threshold. The vertical axis begins at 0.70 to show the fitness cap under reduced-dictionary correction.

Alt text: Three-panel analysis of codebook support and genotype-slot sensitivity. Panel A shows attainment increasing with slots per peak in both conditions while remaining higher under correction. Panel B shows positive attainment with the full correction dictionary, zero attainment after required target words are removed, and intermediate attainment without correction. Panel C shows near-maximal best fitness with full-dictionary correction and without correction, but substantially lower best fitness under reduced-dictionary correction.

Correction differences appear during both approach and completion across the search trajectory

To locate where along the search trajectory the attainment difference emerges, we first tracked entry into a near-target region defined by best fitness of at least 0.90 (Fig. 5). Every correcting run entered this region, compared with 86 percent of runs without correction. Among runs that entered, 74 percent of correcting runs reached an exact match, compared with 42 percent without correction, and median time from first entry to exact match was 70 versus 86 generations. Because correction changes which runs enter the region, the among-entrant completion comparison is conditioned on a post-treatment event and should be interpreted descriptively rather than as an isolated causal effect on completion.

Colonization records show two observed routes to exact attainment. Peaks were reached either by a lineage arriving from the valley remnant or by a lineage maturing in place after entering a peak niche. No colonization occurred through transfer from another peak. Correction was associated with more colonizations through both observed routes, with 168 in-place colonizations and 121 valley arrivals, compared with 50 and 71 without correction.

The correction advantage also persisted when correction was restricted to either side of a pre-correction fitness threshold of 0.90. In the same 20-run close-design basis, full correction produced a mean of 1.20 peaks, correction only at or above 0.90 produced 1.05, correction only below 0.90 produced 1.25, and no correction produced 0.55 (Fig. S2). Both restricted-correction conditions therefore had higher mean attainment than no correction. Contrasts among the three correction conditions were imprecise, however, and their confidence intervals were too wide to distinguish clear differences or support equivalence.

The gating manipulation does not isolate a stage-specific causal effect because changing the threshold alters both which states are eligible for correction and how much correction is delivered. At the 0.90 threshold, correction restricted to states at or above the threshold acted on about one quarter of offspring while retaining most of the attainment observed under full correction. Thresholds of 0.85 and 0.95 changed exposure and outcome together in the same way (Supplementary Information S14). Taken together, the near-target and gating analyses show that correction differences are visible both before and after entry into the near-target region, while leaving unresolved how much of the overall advantage arises at either stage.

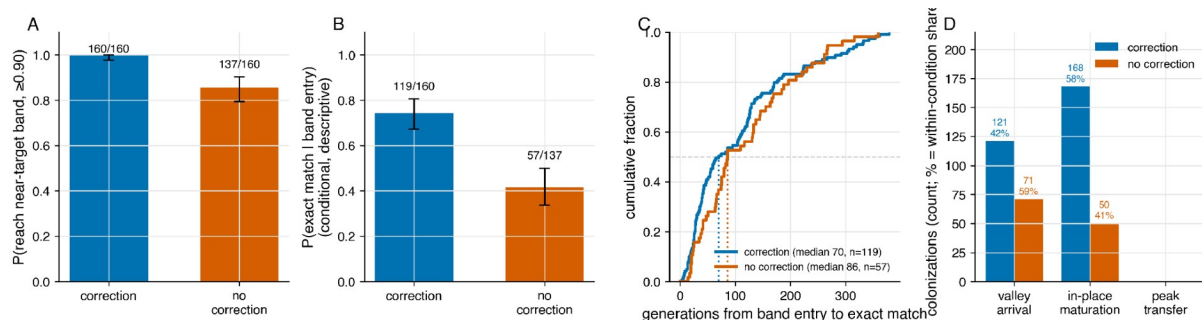


Fig. 5. Correction increases entry into the near-target region, with higher completion among entrants. (A) Proportion of runs reaching a best fitness of at least 0.90, with 95% Wilson intervals. Counts above the bars show the number of runs reaching the threshold over the total number of runs, with 160 runs per condition. (B) Exact matches among runs that reached the threshold. This is a descriptive conditional comparison because threshold entry is itself affected by correction. Counts show exact matches over runs that entered the threshold region. (C) Cumulative distribution of generations from first threshold entry to first exact match among runs that both entered the region and completed a target. Median times of 70 and 86 generations are marked. (D) Colonization counts by source, pooled over 320 runs per condition, with the share of each source shown within condition. Sources are valley arrival, in-place maturation, and transfer from another peak. No colonization occurred through peak transfer.

Alt text: Four-panel process analysis. Correction increases entry into the fitness region at or above 0.90, while exact completion is higher among runs that entered that region. Among runs that entered and later completed a target, median time to completion is 70 generations with correction and 86 without correction. Colonization counts show more valley arrivals and in-place maturations under correction and no colonization through transfer from another peak.

Discussion

Dictionary-based correction increases the number of distinct exact targets reached within a finite search horizon when several target-associated trajectories are retained in parallel. Extended runs show that much of this advantage reflects faster attainment, with populations without correction partly catching up as search continues. Because both conditions remain below the five-peak ceiling, the study does not establish a higher long-run coverage ceiling under correction. The central result is therefore temporal: within a bounded search budget, correction allows more adaptive outcomes to be realized sooner. This timing result suggests a further implication for changing environments: correction may be especially consequential when targets shift before slower search has time to catch up.

The result is more specific than the general claim that constraints can facilitate search. Correction changes where malformed variants are directed, and under the present architecture that redistribution increases finite-horizon target attainment. One plausible interpretation is that variants failing to preserve partial structure are less likely to contribute to trajectories that remain competitive, making the distribution of generated states consequential when search effort is limited. The reduced-mutation comparison helps distinguish this from a simple reduction in copying noise. Across the four sampled multipliers, lowering mutation without correction did not reproduce the attainment observed under correction.

What the admissible set contains is equally important. Removing the words required by the active targets makes the same correction operator harmful, because newly generated target words are then treated as errors and replaced. Codebook content therefore cannot be characterized by size alone. The experiment does not provide a gradual estimate of compatibility between codebook and target structure, but it establishes a sharp boundary condition: correction is advantageous here only when the admissible set preserves components needed by the solutions.

The magnitude of the advantage varies across the sampled regimes, with the largest point estimates at the two intermediate mutation multipliers and smaller average differences in the more widely spaced sampled landscapes. Neither pattern identifies an optimal mutation level or isolates peak spacing as a cause. Both indicate that the correction advantage varies across the noise levels and target arrangements sampled here.

Process analyses place the observed differences on both sides of entry into the near-target region without decomposing the overall effect into causal stages. Correction increases entry into that region, while exact completion is higher among runs that enter it. Restricting correction to states either below or at or above the 0.90 fitness threshold also preserves an advantage over no correction. Because threshold gating changes both exposure to correction and the states exposed, these results do not identify a dominant stage. They show instead that the correction difference is not confined to a single narrow portion of the trajectory.

The architecture and representation define important conditions of the result. Target-specific retention preserves several competing trajectories by design, so the study evaluates correction within a multi-trajectory regime rather than testing whether lineage protection itself moderates the correction effect. Because correction and no-correction runs use the same retention architecture, that architecture does not by itself explain the within-regime treatment contrast. The corrector is also blind to target identity during a run, but its dictionary contains the target vocabulary, and fitness rewards both lexical overlap and character-level similarity. The advantage therefore cannot be attributed to target-aware steering, yet correction and evaluation are structurally compatible in this textual representation.

The design combines two elements emphasized in adjacent literatures. Repair operators, developmental bias, and robustness concern how variation is structured, while niching and multimodal methods concern how competing alternatives are retained (9, 11, 12–14, 18–20). Within an architecture that preserves several trajectories, a target-blind local correction rule can increase the number of exact targets realized within a finite budget, but only when the states favored by correction remain compatible with the solutions being sought.

The textual substrate makes these relationships unusually transparent, but it also limits the scope of the inference. The model uses one language, one dictionary, one fitness function, fixed seeds and targets, four sampled mutation multipliers, a binary codebook intervention, and a target-specific retention architecture. Correction is imposed rather than inherited. Landscape properties other than mean peak distance are not independently decomposed, and the extended runs do not resolve asymptotic coverage. The strongest next tests would therefore vary codebook compatibility gradually, cross correction with alternative retention architectures while matching total search effort, measure realized variation directly, and reproduce the result in a second substrate.

Repair and filtering under bounded search occur in systems well beyond textual replication. Molecular proofreading, pattern-completing memory, and error-correcting codes provide possible analogies, although the present model does not establish that they instantiate the same mechanism (21–26). These results suggest a broader hypothesis: replicating systems may benefit from imperfect error correction when correction is target-blind but its admissible set preserves components needed by multiple solutions. Models in which correction itself evolves can test whether and when selection favors such strategies, and how the favored form of correction changes across environments.

Methods

Replicators and mutation. Each replicator is an English sentence. Initial seeds and targets contain uppercase and lowercase letters, but text is normalized to lowercase for fitness scoring and correction so that case variants are treated alike. Each generation, every replicator produces 10 offspring copied from the parent before mutation. Mutation acts on characters and spaces through substitution, deletion, and insertion. The base probabilities are 0.02 for character substitution, 0.004 for character deletion, 0.004 for character insertion, 0.02 for space deletion, and 0.004 for space insertion. These operations are applied in a fixed order. A single mutation-rate multiplier scales all five probabilities and therefore controls copying noise while the targets and fitness function remain fixed.

Correction. In correction conditions, each word in a mutated offspring is checked against a general English dictionary of 110,879 words. The dictionary already contains every word used in the target sentences, and no target-specific words were added for the experiment (Supplementary Information S2). Words found in the dictionary remain unchanged. A word absent from the dictionary is replaced by a randomly selected dictionary word meeting a combined trigram and edit-distance similarity threshold of 0.8. When no candidate meets the threshold, the original word is retained. The corrector does not reference target identities or target sentences during a run, although its dictionary contains the vocabulary used across the targets.

Fitness. Each replicator is scored against every target. Fitness is the highest resulting similarity score, calculated as the equally weighted mean of length-normalized Levenshtein similarity (17) and set-based word overlap, scaled to the unit interval. In multi-peak runs, the target producing this maximum score is designated the replicator's nearest peak. Using the maximum rather than

an aggregate score prevents strings with weak similarity to several targets from gaining an artificial advantage.

Selection. The single-peak baseline uses the original global-selection rule, which retains the fittest replicators across the full population. Multi-peak runs instead assign each replicator to its nearest peak. Each peak has a fixed number of genotype slots filled by the fittest distinct genotypes assigned to that peak. Replicators that do not fill a slot enter a shared valley remnant with fixed capacity. When the remnant exceeds capacity, it is pruned either by fitness or at random. A peak is counted as reached when one of its slot occupants exactly matches the target. Recorded outcomes include the number of peaks reached, the generation of first attainment for each peak, and the source of each attainment. Because the single-peak and multi-peak designs differ in population structure and the number of lineages retained, their results are not estimates of the same treatment contrast.

Design and pooling. The core design crosses correction with mutation-rate multiplier, valley-pruning rule, peak arrangement, and niche capacity. Mutation-rate multipliers are 0.25, 0.5, 1.0, and 1.5. Standard runs use 20 replications per cell and a maximum of 500 generations. The headline multi-peak estimate pools two independent draws of the close-peak design. The original and regenerated draws share no random seeds and each contribute 160 runs per condition, yielding 320 runs per condition in the pooled analysis.

Implementation equivalence was assessed separately by rerunning selected original cases under the current implementation with their original seeds. Identical outcomes demonstrate behavioral equivalence for those cases but do not contribute independent observations to the experimental analyses. Comparisons across parameter sweeps are descriptive. Figures report 95% confidence intervals, and the headline means are also accompanied by standard errors in the text. No multiplicity correction is applied. Each run is seeded deterministically from its identifier and stored with a manifest containing its parameter configuration, random seed, peak identities, and a hash of the model code.

The authors used Claude Sonnet 5 (Anthropic) and GPT-5.6 Sol (OpenAI) for editorial and analytic assistance under author direction, including code drafting, manuscript and supplementary drafting and editing, assistance with code used to render figures from analysis output, and analysis review. Figures and numerical results were produced from documented analysis outputs using author-reviewed scripts rather than generated directly by a generative model. The authors reviewed, revised, and take responsibility for all content.

Data availability

Run manifests and summary data supporting the findings are available at <https://github.com/ericcgldstone-maker/bfg-corrective-replicator>. Full per-generation output is archived on the Open Science Framework at <https://doi.org/10.17605/OSF.IO/K2PH7>.

Code availability

The simulation and analysis code used for this study is available at <https://github.com/ericcgldstone-maker/bfg-corrective-replicator>. The original Java implementation on which the current model is based is available at <https://github.com/lardener/BFG>.

Acknowledgements

This study received no funding.

Author contributions

Conceptualization: M.E.B., E.G., J.F. **Methodology:** J.F., E.G., M.E.B. **Software:** E.G., J.F. **Formal analysis:** E.G., M.E.B. **Investigation:** E.G., M.E.B. **Visualization:** E.G. **Writing, original draft:** E.G., M.E.B. **Writing, review and editing:** all authors. **Supervision:** M.E.B.

Competing interests

The authors declare no financial or non-financial competing interests.

References

1. Dawkins, R. (1976). *The selfish gene*. Oxford University Press.
2. Hull, D. L. (1988). *Science as a process*. University of Chicago Press.
3. Wright, S. (1932). The roles of mutation, inbreeding, crossbreeding, and selection in evolution. *Proceedings of the Sixth International Congress of Genetics*, 1, 356–366.
4. de Visser, J. A. G. M., & Krug, J. (2014). Empirical fitness landscapes and the predictability of evolution. *Nature Reviews Genetics*, 15(7), 480–490.
5. Poelwijk, F. J., Kiviet, D. J., Weinreich, D. M., & Tans, S. J. (2007). Empirical fitness landscapes reveal accessible evolutionary paths. *Nature*, 445(7126), 383–386.
6. Kauffman, S. A. (1993). *The origins of order: Self-organization and selection in evolution*. Oxford University Press.
7. Levinthal, D. A. (1997). Adaptation on rugged landscapes. *Management Science*, 43(7), 934–950.
8. van Nimwegen, E., Crutchfield, J. P., & Huynen, M. (1999). Neutral evolution of mutational robustness. *Proceedings of the National Academy of Sciences*, 96(17), 9716–9720.
9. Wagner, A. (2008). Robustness and evolvability: A paradox resolved. *Proceedings of the Royal Society B: Biological Sciences*, 275(1630), 91–100.
10. Brashears, M. E., & Gladstone, E. (2016). Error correction mechanisms in social networks reduce accuracy and encourage innovation. *Social Networks*, 44, 22–35.
11. Gerhart, J., & Kirschner, M. (2007). The theory of facilitated variation. *Proceedings of the National Academy of Sciences*, 104(suppl. 1), 8582–8589.
12. Salcedo-Sanz, S. (2009). A survey of repair methods used as constraint handling techniques in evolutionary algorithms. *Computer Science Review*, 3(3), 175–192.

13. Uller, T., Moczek, A. P., Watson, R. A., Brakefield, P. M., & Laland, K. N. (2018). Developmental bias and evolution: A regulatory network perspective. *Genetics*, 209(4), 949–966.
14. Li, X., Epitropakis, M. G., Deb, K., & Engelbrecht, A. (2017). Seeking multiple solutions: An updated survey on niching methods and their applications. *IEEE Transactions on Evolutionary Computation*, 21(4), 518–538.
15. Dawkins, R. (1986). *The blind watchmaker*. Norton.
16. Wilke, C. O., & Adami, C. (2002). The biology of digital organisms. *Trends in Ecology & Evolution*, 17(11), 528–532.
17. Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8), 707–710.
18. Goldberg, D. E., & Richardson, J. (1987). Genetic algorithms with sharing for multimodal function optimization. In J. J. Grefenstette (Ed.), *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms* (pp. 41–49). Lawrence Erlbaum Associates.
19. Pugh, J. K., Soros, L. B., & Stanley, K. O. (2016). Quality diversity: A new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 3, 40.
20. Lehman, J., & Stanley, K. O. (2011). Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary Computation*, 19(2), 189–223.
21. Kunkel, T. A. (2004). DNA replication fidelity. *Journal of Biological Chemistry*, 279(17), 16895–16898.
22. Hopfield, J. J. (1974). Kinetic proofreading: A new mechanism for reducing errors in biosynthetic processes requiring high specificity. *Proceedings of the National Academy of Sciences*, 71(10), 4135–4139.
23. Rolls, E. T. (2013). The mechanisms for pattern completion and pattern separation in the hippocampus. *Frontiers in Systems Neuroscience*, 7, 74.
24. Shannon, C. E. (1948). A mathematical theory of communication. *Bell System Technical Journal*, 27(3), 379–423.
25. MacKay, D. J. C. (2003). *Information theory, inference, and learning algorithms*. Cambridge University Press.
26. Hamming, R. W. (1950). Error detecting and error correcting codes. *Bell System Technical Journal*, 29(2), 147–160.

Supplementary Information

A corrective constraint increases multi-target attainment under finite adaptive search

Brashears, Gladstone, and Ferrer

S1. Overview

The simulation evolves a population of textual replicators, each represented by an English sentence, toward target sentences that define peaks in a fitness landscape. Each generation proceeds through reproduction, mutation, optional correction, fitness evaluation, and selection. These operations are implemented as separate modules.

The simulation is implemented in Python as a reimplement of the original Java model developed by Brashears, Gladstone, and Ferrer (available at <https://github.com/lardener/BFG>). The current implementation follows the documented model specification and preserves the original correction logic and overall sequence of reproduction, mutation, fitness evaluation, and selection. Implementation differences from the Java code are described where relevant below, including the mutation character pool in S3 and the word-overlap rule in S4. Selected original cases were rerun with their original seeds to assess behavioral equivalence (S10). The current implementation adds the multi-peak selection procedure described in S6. Experimental conditions are defined through parameter settings rather than changes to the core simulation code.

The main simulation is implemented in `analysis/bfg_stage2.py`. The single-peak baseline uses the original global-selection procedure, while the multi-peak experiments use target-specific retention. Experiment scripts, including `analysis/run_stage2.py`, construct the peak sets, assign deterministic random seeds, execute the runs, and write per-generation and per-run outputs together with a reproducibility manifest. The code is available at <https://github.com/ericcgldstone-maker/bfg-corrective-replicator>.

S2. Replicators, seeds, targets, and dictionary

A replicator is a text string. Each run begins with the same set of 16 fixed seed sentences. The environment contains 100 fixed target sentences, of which a selected subset defines the peaks used in a given experiment (S8).

Seeds and targets contain uppercase and lowercase letters. Text is normalized to lowercase for fitness scoring and correction so that case variants are treated as equivalent. This normalization does not alter the original strings stored in the input data.

Correction uses a general English dictionary containing 110,879 words. All 633 distinct words appearing in the 100 target sentences were already present in this dictionary, and no target words were added for these experiments. Target words are therefore available to the correction procedure, although the corrector does not reference target identities during a run. The original Java implementation instead added the scrubbed target words to the dictionary when loading it. For the target set studied here, the two procedures produce the same codebook because every target word is already present in the general list. A trigram index supports efficient retrieval of candidate replacements (S5).

S3. Reproduction and mutation

Each retained replicator produces 10 offspring per generation in all reported experiments, and each offspring begins as an exact copy of its parent before mutation and optional correction. Mutation introduces variation at the level of characters and spaces. Five operators act with fixed base probabilities. A mutation-rate multiplier scales all five probabilities and controls copying noise, while the targets and fitness function remain fixed across multiplier conditions.

Operator	Base probability
Character substitution	0.02
Character deletion	0.004
Character insertion	0.004
Space deletion	0.02
Space insertion	0.004

The operators are applied in a fixed order as the string is scanned from left to right. At each character position, deletion is evaluated first. If the character remains, substitution may replace it with a character drawn uniformly from a fixed pool containing the 26 lowercase letters, a hyphen, and an apostrophe. Replicator text is lowercased before mutation, so uppercase characters do not enter this pool. Character insertion draws from the same pool and is evaluated next, followed by space deletion and insertion.

The original Java implementation drew from a 54-character pool containing both letter cases, which made each letter about twice as likely as either punctuation mark. Because text is lowercased for scoring and correction, uppercase and lowercase letters collapse to the same scored form, but the smaller character pool changes the relative probability of punctuation substitutions and insertions. The current implementation nevertheless reproduces the single-peak pattern first obtained with the original implementation (S16). This pattern-level replication does not make the two mutation operators identical.

More than one operator can act at the same position. Some operations can therefore cancel, such as deleting and then reinserting a space. The realized mutation rate is consequently slightly lower than the sum of the nominal operator probabilities. Before each run, the specified mutation-rate multiplier is applied to every base probability.

S4. Fitness

A replicator's fitness is its similarity to a target, scaled to the unit interval. Similarity combines two equally weighted components:

1. **Character similarity:** one minus the Levenshtein edit distance (1) between the replicator and target, normalized by the length of the longer string.
2. **Word-set overlap:** the number of distinct words shared by the replicator and target, divided by the number of words in the longer string.

The final fitness score is the mean of these two components. Word overlap is set-based, so each distinct word counts once. This follows the documented methods specification used as the canonical rule, although the original Java implementation used a multiset.

In multi-peak runs, each replicator is scored against every active peak. Its highest score determines both its fitness and its nearest peak. Using the maximum rather than an aggregate score prevents weak similarity to several targets from producing an artificial fitness advantage.

S5. Correction

In the correction condition, each mutated offspring passes through a dictionary-based corrector one word at a time. Words found in the dictionary remain unchanged. For a word absent from the dictionary, the corrector retrieves candidates meeting a combined trigram (2) and Levenshtein similarity threshold of 0.8. When one or more candidates qualify, one is selected uniformly at random. When none qualify, the original nonword is retained.

The corrector does not reference target identities or target sentences during a run. It maps malformed words toward valid dictionary entries rather than toward a specified peak. The dictionary does, however, contain every word used in the targets, so the admissible set contains the components of all target solutions. The codebook ablation in S13 tests the importance of that support. A null corrector disables correction and defines the no-correction condition.

S6. Selection

Single-peak baseline. The original global-selection rule pools all offspring, ranks them by fitness, and retains the 1,000 fittest. Because selection operates across all targets at once, the population converges on whichever target is easiest to reach. This global competition leaves one dominant peak.

Multi-peak selection. The multi-peak design replaces global selection with target-specific retention. The procedure was adapted from a per-target bucket mechanism present but disabled in the original Java code. Each offspring is assigned to its nearest peak. Each peak has a fixed number of slots, with 10 used by default, and each slot retains a distinct genotype. The fittest genotypes assigned to a peak fill its slots, and a fitter arrival can displace the least fit occupant.

Offspring that do not secure a peak slot enter a shared pool of between-peak replicators called the valley remnant. The remnant has a fixed capacity and allows lineages outside the protected peak niches to persist across generations. When the remnant exceeds capacity, it is pruned either by fitness, retaining the replicators closest to a peak, or at random. Both pruning rules are included as experimental conditions. Replicators retained in neither a peak slot nor the valley remnant are removed.

Because each peak holds up to its slot capacity and the valley remnant capacity is fixed at 200, the total retained population rises with the number of slots per peak. At 5, 10, 20, and 40 slots, the retained population is 225, 250, 300, and 400 replicators, respectively, and the number of offspring produced each generation rises in proportion. The niche-capacity experiment therefore

varies protected retention together with total reproductive search effort rather than isolating capacity alone. Correction and no-correction conditions are compared at the same capacity, so their within-capacity contrast remains matched on this scaling.

S7. Colonization and attainment measures

A peak is colonized when one of its assigned slot occupants reaches the exact-match threshold for that target. The exact-match threshold is a fitness of 0.99999, used as a floating-point tolerance for exact identity. Under the fitness function, a score at or above this value is reachable only at Levenshtein distance zero, so colonization requires an exact string match. Colonization is recorded as a first-attainment event and does not require the exact match to remain occupied for the rest of the run.

The following outcomes are recorded for each run:

- **Peaks colonized:** the cumulative number of distinct peaks that reach an exact match within the run. This is the primary target-attainment measure.
- **Generation of first colonization:** the first generation at which each peak is reached, together with the time required to reach specified numbers or fractions of peaks.
- **Colonization source:** whether the first exact match for a peak arrived from the valley remnant, matured in place from a lineage already assigned to that peak, or transferred from another peak. This measure distinguishes direct valley arrival, local completion, and peak transfer.

S8. Peak arrangements

Peak arrangements are fixed and recorded subsets of the 100 targets selected to produce different target configurations.

- **Close arrangement:** an anchor target is grouped with its nearest neighbors by edit distance. Several peaks are reachable within the standard horizon, making this the headline multi-peak design.
- **Distance ladder:** six arrangements span mean pairwise edit distances of approximately 46 to 72, constructed through a seeded search over subsets of five targets grouped by mean pairwise distance. This ladder supports the valley-pruning comparison in S11.

Exact target indices for every arrangement are recorded in the run manifests. The replicated landscape analysis in main-text Fig. 3B uses three independently sampled peak sets at each of three mean-distance levels rather than one landscape at each level of the distance ladder.

S9. Experimental design

The core design crosses correction with mutation-rate multiplier, valley-pruning rule, peak arrangement, and niche capacity. Mutation-rate multipliers are 0.25, 0.5, 1.0, and 1.5. Valley remnants are pruned either by fitness or at random, and the capacity experiment varies the number of genotype slots per peak from 5 to 40. Standard experiments use 20 replications per cell and a maximum of 500 generations, with exceptions specified in their respective sections. Table S1 summarizes the principal parameter values.

Table S1. Model and experimental parameters.

Parameter	Value	Role
Offspring per parent	10	Reproduction
Character substitution probability	0.02	Mutation
Character deletion probability	0.004	Mutation
Character insertion probability	0.004	Mutation
Space deletion probability	0.02	Mutation
Space insertion probability	0.004	Mutation
Mutation-rate multiplier	0.25, 0.5, 1.0, 1.5	Copying noise
Correction similarity threshold	0.8	Candidate replacement threshold
Dictionary size	110,879 words	Correction codebook
Target words in dictionary	All target words included by default	Codebook support
Fitness components	Character similarity and word-set overlap, equally weighted	Fitness
Target assignment	Maximum similarity across active peaks	Nearest peak
Number of active peaks	5 unless otherwise specified	Target environment
Slots per peak	10 by default, varied from 5 to 40	Niche capacity

Valley remnant capacity	200	Between-peak retention
Valley-pruning rule	Fitness or random	Remnant selection
Exact-match threshold	0.99999	Colonization criterion
Standard generation limit	500	Run length
Extended horizon	1,500 generations	Timing analysis
Standard replications per cell	20	Stochastic replication
Initial seeds	16 fixed sentences	Initialization
Candidate targets	100 fixed sentences	Environment

S10. Reproducibility and verification

Each run is seeded deterministically from its identifier. Every run set includes a manifest recording the full configuration, random seeds, peak identities and strings, seed sentences, dictionary size, and an MD5 hash of the model code. The manifest and corresponding input files permit exact reproduction of each run.

Code-version verification. Some datasets were generated with earlier versions of the Python model file. Selected original correction runs were repeated under the current implementation using their original random seeds and produced identical outcomes, demonstrating behavioral equivalence for the verified cases.

The original and regenerated draws of the headline multi-peak design share no random seeds and are treated as independent draws from the same process. Each contributes 160 runs per condition, yielding 320 runs per condition after pooling. Code equivalence and statistical independence are distinct claims. Reproducing an earlier run with the same seed verifies implementation consistency but does not create an independent observation.

Audit. All reported numerical results were recalculated from the raw per-run output and checked against the manuscript within rounding. The complete audit record is available in [docs/BFG_Audit_Jul9.md](#) in the code repository.

Output. Each run writes a per-generation file, a per-run summary, and a manifest containing the configuration and reproducibility information described above. These files generate every figure and numerical summary in the manuscript and supplement. Field-level descriptions accompany

the code, and full per-generation output is archived on the Open Science Framework at <https://doi.org/10.17605/OSF.IO/K2PH7>.

Statistical procedures. Confidence intervals for mean peaks colonized, mean fitness, and their time courses are normal-approximation intervals, computed as the mean plus or minus 1.96 standard errors, where the standard error is the sample standard deviation divided by the square root of the number of runs. Proportions in Fig. 5 use Wilson score intervals. Paired contrasts in the gating analysis use the mean of within-seed paired differences plus or minus 1.96 standard errors of those differences. Differences between correction and no correction for individual landscapes and for valley pruning combine the two condition standard errors as the square root of the sum of their squares. In the pooled headline analysis, correction and no-correction runs do not share random seeds, so the treatment difference of 0.52 peaks uses this unpaired combination and has a 95 percent confidence interval from 0.43 to 0.62. No multiplicity correction is applied, and comparisons across parameter sweeps are descriptive.

S11. Extended horizon and valley-remnant selection

Extended horizon. The close-peak design was run to 1,500 generations to distinguish the timing of the correction advantage from its longer-run magnitude.

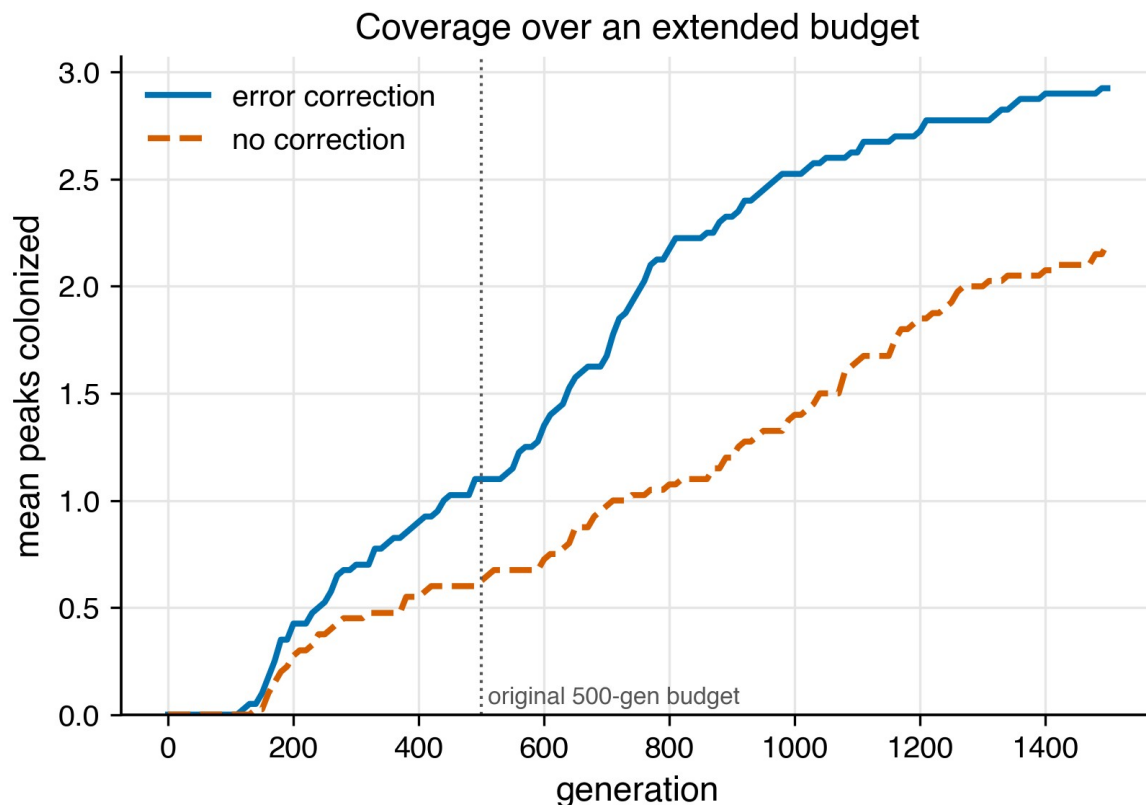


Fig. S1. Target attainment over an extended horizon. Mean cumulative peaks colonized as a function of generation in the close-peak design, with and without correction. The analysis pools mutation-rate multipliers 0.5 and 1.0 across 80 runs and extends the horizon to 1,500 generations. The dotted line marks the standard 500-generation horizon. Correction retains an advantage

throughout the observed interval and reaches a mean of 2.92 peaks, compared with 2.20 without correction, at generation 1,500. The difference reaches about one target near generation 1,000 and narrows to about 0.7 by generation 1,500 as the no-correction condition partly catches up. Both conditions remain below the five-peak ceiling, so the analysis does not establish their eventual coverage or relative asymptotes. Peaks colonized is cumulative. When a run ends before generation 1,500, its last observed count is carried forward rather than replaced with zero.

Alt text: Mean cumulative peaks reached over 1,500 generations. Correction remains above no correction throughout the observed interval. The difference grows until roughly generation 1,000 and then narrows as the no-correction condition partly catches up, while both conditions remain below the five-peak ceiling.

Valley-remnant pruning. The valley remnant is pruned when it exceeds capacity. Fitness pruning retains the replicators closest to an active peak, while random pruning samples without regard to fitness. Across the six landscapes in the distance ladder, with correction enabled and 40 runs per pruning rule at each distance, fitness pruning produced attainment equal to or greater than random pruning at every tested distance. The estimated difference increased from about 0.18 peaks at the closest spacing, with a 95% confidence interval from minus 0.15 to 0.51, to about 0.60 at the widest spacing, with an interval from 0.44 to 0.76. Mean pairwise distance was manipulated without holding other properties of each target set constant. The pattern is therefore consistent with a larger advantage for fitness pruning in the more widely spaced landscapes sampled here, but it does not isolate distance as the cause.

S12. Robustness and sensitivity analyses

These analyses test whether the target-attainment result depends on the attainment criterion, the correction-similarity threshold, lower mutation without correction, or one target arrangement. Unless otherwise noted, they use the close-peak design at mutation-rate multiplier 1.0 with fitness pruning, 20 replications per condition, 500 generations, deterministic seeds, and run manifests. The replicated-landscape analysis uses 12 runs per condition for each sampled landscape. The extended-horizon analysis in Fig. S1 addresses how much of the finite-horizon advantage reflects faster attainment.

Relaxed colonization criterion. We redefined colonization as reaching either 0.95 or 0.90 of maximum fitness rather than an exact match. The estimated correction advantage remains positive under both relaxed criteria and is larger in absolute terms than under exact matching.

Colonization criterion	Correction	No correction	Difference
Exact match	1.20	0.55	0.65
0.95 of maximum fitness	2.00	1.00	1.00
0.90 of maximum fitness	3.55	1.85	1.70

The positive attainment difference therefore does not depend on requiring exact completion within these tested criteria. The three criteria are nested measures of the same runs and should not be treated as independent replications.

Correction-similarity threshold. The default dictionary-similarity threshold of 0.80 was varied to 0.70 and 0.90. The estimated difference is unchanged between 0.70 and 0.80 and smaller at 0.90.

Similarity threshold	Correction	No correction	Difference
0.70	1.20	0.55	0.65
0.80	1.20	0.55	0.65
0.90	0.85	0.55	0.30

The correction advantage is unchanged between the sampled thresholds of 0.70 and 0.80 and smaller at 0.90. With only three sampled thresholds, this analysis does not identify an optimal correction threshold.

Reduced mutation without correction. Mutation rate and correction are both manipulated parameters, so we asked whether lowering mutation alone reproduces the correction advantage. Using the headline close design, we compared correction at the default mutation multiplier of 1.0 with the no-correction condition at each of the four sampled multipliers.

Mutation multiplier	Correction	No correction
0.25	0.75	0.45
0.5	1.16	0.64
1.0	1.11	0.40
1.5	0.59	0.03

Within the four sampled multipliers, the highest observed mean attainment without correction was 0.64 at multiplier 0.5, compared with 1.11 under correction at multiplier 1.0. Every no-correction mean lies below correction at multiplier 1.0. Because the highest no-correction condition was identified after inspecting all four outcomes, we do not attach a confidence interval to this across-multiplier comparison. Only four multipliers were sampled, so the analysis does not identify an optimal mutation rate. Within the tested range, reducing mutation without correction did not reproduce the attainment observed under correction.

Replicated landscapes by distance. To reduce dependence on a single target set, we sampled three independent landscapes at each of three mean distances between peaks. Point estimates favored correction in all nine landscapes, and the average estimated difference was smaller at wider mean spacing. Several confidence intervals for individual landscapes included zero.

Mean distance	Landscape	Correction	No correction	Difference
approximately 50	a / b / c	1.83 / 1.08 / 0.83	1.08 / 0.50 / 0.33	0.75 / 0.58 / 0.50
approximately 60	a / b / c	0.58 / 0.83 / 0.83	0.33 / 0.42 / 0.67	0.25 / 0.42 / 0.17
approximately 70	a / b / c	0.58 / 0.75 / 0.58	0.50 / 0.67 / 0.42	0.08 / 0.08 / 0.17

The target sets differ in properties other than mean pairwise distance, so the analysis does not isolate spacing as the cause of the smaller average difference. It shows that the estimated correction advantage remained positive across the nine sampled landscapes and was smaller, on average, in the more widely spaced sets.

S13. Codebook support for target components

The default dictionary contains every word used in the target sentences, so the correction codebook contains the components of all target solutions. To test the importance of that support, we removed the 51 distinct words used in the five active target sentences and rebuilt the trigram index. All other features of the design were held fixed, including the peak set, random seeds, mutation-rate multiplier of 1.0, fitness pruning, and 20 replications per condition.

Removing the target words reduced mean attainment under correction from 1.20 peaks to zero, compared with 0.55 under no correction. The no-correction runs were identical seed by seed to their counterparts in the full-dictionary experiment, confirming that the manipulation altered only the correction path.

The reduced dictionary changes the corrector in two ways. It removes codebook support for the target words and causes target words generated during search to be treated as errors. When mutation generates an excluded target word, the corrector replaces it with another dictionary entry. Reduced-dictionary correction therefore actively redirects lineages away from the excluded solution components.

The intervention also imposes a fitness cap. Mean best fitness under reduced-dictionary correction was about 0.76, and no run reached 0.90. Mean best fitness without correction was about 0.98, and every run reached 0.90. The reduced-dictionary condition therefore produces zero colonization under both exact matching and the relaxed 0.90 criterion.

Criterion	Full-dictionary correction	Reduced-dictionary correction	No correction
-----------	-------------------------------	----------------------------------	------------------

Exact match	1.20	0.00	0.55
0.90 of maximum fitness	3.55	0.00	1.85

This intervention tests a boundary condition rather than decomposing the correction effect. Removing target words both eliminates codebook support for required solution components and causes those components to be rejected when they appear. The contrast therefore does not estimate a gradual effect of codebook compatibility. It establishes the narrower result that, under the present correction operator, preserving required target components is a condition of the observed advantage, while excluding those components can make correction harmful.

The analysis is regenerated from raw output by [analysis/analyze_dict_ablation.py](#).

S14. Fitness-gated correction

To examine where along the search trajectory the correction difference appears, we applied correction only to offspring whose pre-correction fitness fell on one side of a threshold of 0.90. The threshold changes which portions of the trajectory are exposed to correction. It is not assumed to distinguish separate biological or computational processes.

The gating decision was based on each mutated offspring before correction. All other features of the headline close-peak design were unchanged, including mutation-rate multiplier 1.0, fitness pruning, 20 replications per condition, and shared random seeds across paired conditions. The no-correction arms were identical to the corresponding controls in the full-correction experiment, confirming that the manipulation altered only the correction path.

Full correction produced a mean of 1.20 peaks. Correction applied only at or above 0.90 produced 1.05, correction applied only below 0.90 produced 1.25, and no correction produced 0.55 (Fig. S2). Both restricted-correction conditions therefore had higher mean attainment than no correction.

Paired comparisons among the three correction conditions were imprecise. The mean difference between full and above-threshold correction was 0.15 peaks, with a 95% confidence interval from minus 0.18 to 0.48. The difference between full and below-threshold correction was minus 0.05, with an interval from minus 0.38 to 0.28. The difference between below-threshold and above-threshold correction was 0.20, with an interval from minus 0.32 to 0.72. These comparisons do not identify a clear difference among the correction conditions, and the intervals are too wide to support equivalence.

We repeated the intervention at thresholds of 0.85 and 0.95 and recorded correction exposure, defined as the fraction of offspring passed to the corrector under each gate.

Threshold	Above only	Below only	Above exposure	Below exposure
0.85	1.50	1.00	41%	55%

0.90	1.05	1.25	24%	71%
0.95	0.65	1.30	9%	87%

As the threshold rises, above-threshold correction applies to fewer offspring and its estimated outcome declines. Below-threshold correction applies to more offspring and its estimated outcome increases. Threshold position and correction exposure therefore change together, so the design does not identify their independent effects.

Correction restricted to moderate- and high-fitness offspring can retain much of the attainment observed under full correction while acting on a minority of offspring. At the 0.90 threshold, above-threshold correction produces 1.05 peaks while acting on 24 percent of offspring. At the 0.85 threshold, it produces 1.50 peaks while acting on 41 percent. By contrast, correction restricted to the narrow region above 0.95 acts on 9 percent of offspring and produces 0.65 peaks, close to the no-correction mean.

Taken together, the gating results show that the correction advantage can persist when exposure is restricted to states on either side of the 0.90 threshold. Because changing the gate also changes correction exposure, the experiment does not identify a stage-specific causal effect or determine how much of the overall advantage arises at different portions of the trajectory.

The analysis is regenerated from raw output by [analysis/analyze_gate.py](#).

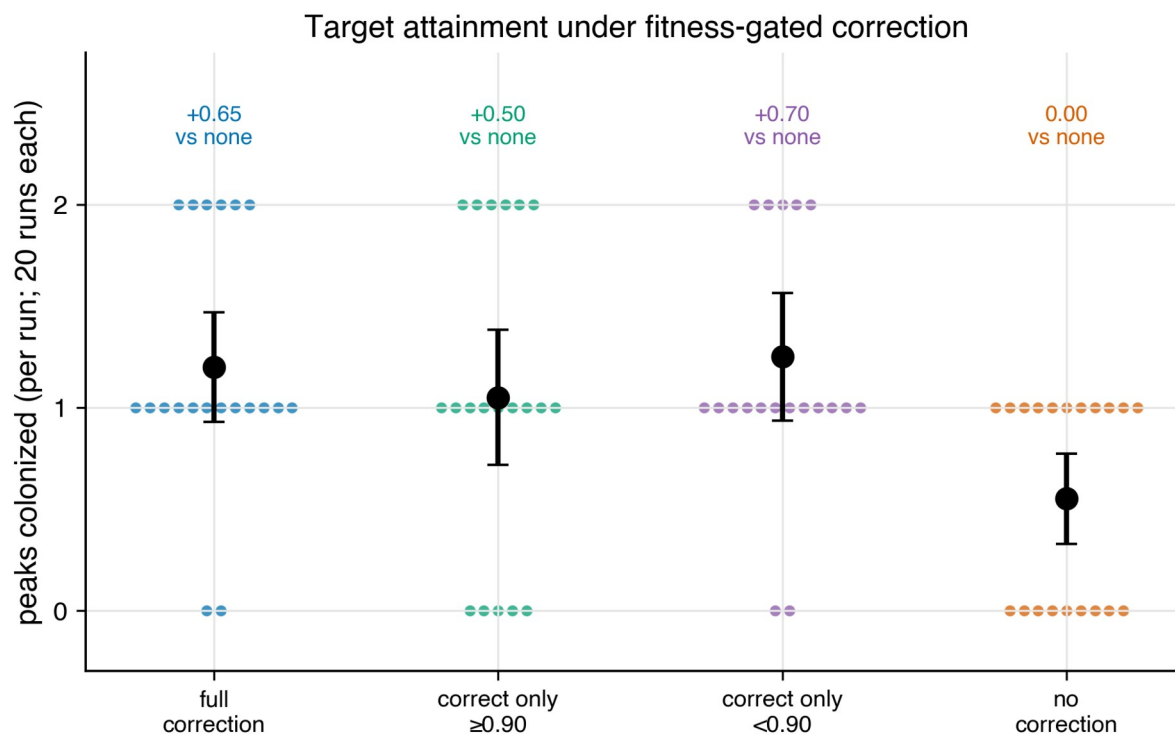


Fig. S2. Fitness-gated correction. Peaks colonized per run under full correction, correction applied only to offspring at or above the 0.90 pre-correction fitness threshold, correction applied only below that threshold, and no correction. Each point represents one of 20 runs per condition. Black markers show the mean and 95% interval. Both restricted-correction conditions have higher mean attainment than no correction, while paired contrasts among the correction conditions remain imprecise.

Alt text: Peaks reached per run under full correction, correction restricted to offspring at or above the 0.90 pre-correction fitness threshold, correction restricted to offspring below the threshold, and no correction. Both restricted-correction conditions have higher mean attainment than no correction.

S15. Experiment inventory and interpretation

Experiment inventory. Each analysis is mapped to its manipulated factors, landscapes, replications per condition, outcome, figure, and data folder. All experiments use deterministic seeds and reproducibility manifests.

Analysis	Manipulated factors	Landscapes	Runs per condition	Outcome	Figure / section	Data folder
Single-peak baseline	Correction, mutation multiplier	Full target environment with global selection	20	Mean fitness	Fig. S4	stage1_baseline
Multi-peak headline	Correction, mutation multiplier	One close cluster	320, pooled from 160 plus 160	Peaks colonized	Fig. 2	stage2_run2_fixed_v2
Extended horizon	Correction	One close cluster	80, pooled across multipliers 0.5 and 1.0	Peaks colonized by generation	Fig. 3A, Fig. S1	asymptotic
Replicated landscapes	Correction, mean distance	Nine landscapes, three per distance	12	Peaks colonized	Fig. 3B	multiland

Valley-remnant pruning	Fitness or random pruning	Six-landscape distance ladder	40	Peaks colonized	S11	<code>stage2_distance</code>
Niche capacity	Correction, slots per peak	One close cluster	approximately 40 per level	Peaks colonized	Fig. 4A	<code>stage2_slots_v2</code>
Codebook ablation	Full-dictionary, reduced-dictionary, or no correction	One close cluster	20	Peaks colonized, best fitness	Fig. 4B, Fig. 4C, S13	<code>dict_ablation_relaxed</code>
Fitness-gated correction	Correction above or below 0.90	One close cluster	20	Peaks colonized	Fig. S2	<code>gate_above, gate_below</code>
Relaxed colonization	Criterion 0.90 or 0.95	One close cluster	20	Peaks colonized	S12	<code>colon_0.90, colon_0.95</code>
Correction threshold	Similarity threshold 0.70, 0.80, or 0.90	One close cluster	20	Peaks colonized	S12	<code>simthresh_0.70, simthresh_0.80, simthresh_0.90</code>

Model-to-construct map. Each computational element is mapped to its intended interpretation and the condition that limits that interpretation.

Model element	Computational role	Intended interpretation	Scope condition
Dictionary	Defines the admissible lexical set	Corrective codebook	Must preserve viable solution components for the observed advantage

Peak slots	Retain target-associated genotypes	Protected niches	Multi-target attainment is studied within an architecture that preserves competing trajectories
Mutation multiplier	Scales mutation probabilities	Copying noise	Does not alter targets or the fitness function
Exact match	Records first colonization	Target completion	Does not imply persistent occupation
Valley remnant	Retains replicators outside peak slots	Between-peak explorer pool	Capacity is externally imposed

S16. Original single-peak result

The single-peak result was first obtained with the original implementation of the model, before target-specific selection was added. Under global selection over the full target environment, dictionary-based correction accelerated attainment of a single target and preserved higher fitness as the mutation rate increased (Fig. S3). The current implementation reproduces the same qualitative pattern under global selection (Fig. S4).

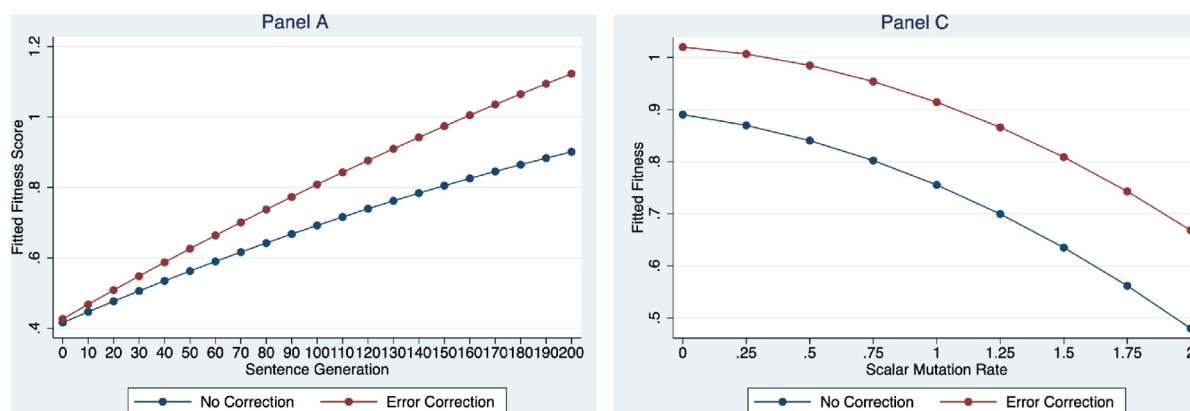


Fig. S3. Original single-peak result. Correction compared with no correction in the original implementation of the model. Left: fitted fitness over sentence generations, showing a faster increase under correction. Right: fitted fitness across the mutation-rate scalar, showing that correction preserves higher fitness as the mutation rate rises. The curves are fitted values from the original analysis and extend slightly above the unit fitness scale for that reason.

Alt text: Two-panel single-peak result from the original implementation. The left panel shows a faster increase in fitted fitness over generations with correction. The right panel shows higher fitted fitness under correction as the mutation-rate scalar increases, while fitted fitness without correction declines.

Correction accelerates the single-peak climb. We next evaluated the single-peak baseline under the current Python implementation (Fig. S4). Correcting populations reached high fitness sooner at every sampled mutation rate, and the difference was larger at higher copying noise. At low and moderate mutation rates, the two conditions converged to similar final fitness within 500 generations. At the highest sampled rate, populations without correction stalled at substantially lower fitness, while correcting populations continued to improve. The current implementation therefore reproduces the qualitative baseline pattern: correction accelerates attainment of high fitness under global selection and preserves higher fitness at the highest sampled mutation rate.

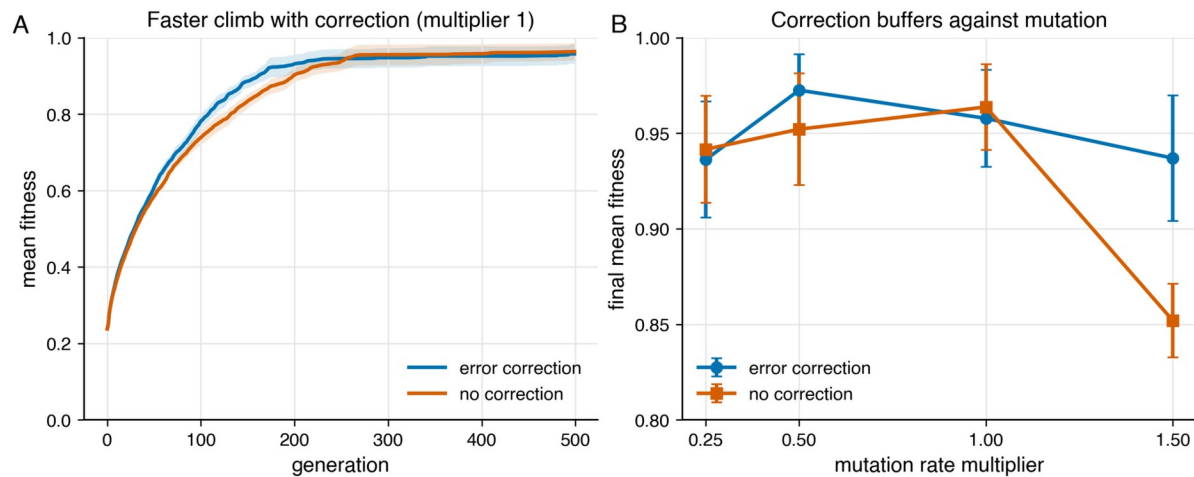


Fig. S4. Single-peak baseline. The analysis uses the global-selection architecture over the full target environment, with a population of 1,000, 10 offspring per replicator, 20 replications per condition, and 500 generations. (A) Mean fitness over generations for populations with and without correction at mutation-rate multiplier 1.0, with 95% confidence bands. Correction reaches high fitness sooner, after which the conditions converge. (B) Final mean fitness across four mutation-rate multipliers, with 95% confidence intervals. Final fitness is similar at low and moderate rates, while correction preserves higher fitness at the highest sampled rate.

Alt text: Two-panel comparison of single-peak adaptation with and without correction. Panel A shows mean fitness increasing faster under correction before the two conditions converge near maximum fitness. Panel B shows similar final fitness at the lower mutation-rate multipliers, while final fitness without correction declines at the highest sampled multiplier and remains higher under correction.

References

1. Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8), 707–710.
2. Kondrak, G. (2005). N-gram similarity and distance. In *String processing and information retrieval* (Lecture Notes in Computer Science, Vol. 3772, pp. 115–126). Springer.