

Algorithmic Screening: A Concept Note on Systematic Discovery of Better Algorithms

Bjørn Remseth
Email: la3lma@gmail.com

Abstract—This note presents an initial exploration of *algorithmic screening*, an approach inspired by drug discovery. The premise is straightforward: many software systems rely on long-established algorithms whose performance, complexity, or structural assumptions may no longer match contemporary computational environments. By combining high-volume experimentation, representation-level search, and modern automated synthesis techniques, one may systematically surface algorithmic variants and refinements that offer meaningful improvements. Even modest gains could accumulate across large-scale systems in ways that are both practically valuable and conceptually enlightening. The idea is simple, yet carries a quiet but genuine promise.

version: 2025-11-20-14:39:18-CET-083be39-main

I. INTRODUCTION

Modern software ecosystems contain substantial volumes of code that work adequately but rest upon algorithms chosen decades ago under constraints that have since shifted. Hardware is different, memory hierarchies are different, problem sizes have grown, and new mathematical tools and learned representations have emerged. Yet much deployed software remains, quite literally, *crusty*: well-used, trusted, and deeply embedded, but rarely revisited at the algorithmic level.

The central question of this note is disarmingly simple:

What if we systematically screened algorithmic alternatives, at scale, to discover improvements to long-established computational methods?

The idea echoes drug discovery: given a biological target, one generates and evaluates vast numbers of molecular candidates, searching for unexpected and effective treatments. Here, the “disease” is computational complexity or algorithmic inefficiency; the “candidates” are algorithmic variants, parameterizations, restructurings, or representational changes. One tests them against tasks that reflect real workloads, and one does this not as an artisanal process, but as a structured and automated endeavour.

The proposal is straightforward, and yet it carries an understated sense of promise. We know that the past decade has already shown the value of large-scale automated search in machine learning; we know that many algorithmic choices, once considered fixed, have benefited from reinvention. Gaussian Splatting [1] replaced decades of established 3D rendering pipelines; attention mechanisms [2] supplanted recurrent

architectures that had dominated sequence modeling for years. It is therefore not unreasonable to imagine that a systematic approach could yield improvements that are both intellectually satisfying and practically useful.

II. CONCEPT

The basic structure mirrors established scientific workflows:

- 1) Identify an algorithmic target: a sorting routine, a graph traversal, a numerical solver, a compression stage, or any computational routine whose performance is critical.
- 2) Collect representative workloads: ideally real-world data, suitably anonymised or synthetically reproduced to preserve relevant performance characteristics.
- 3) Generate algorithmic candidates: parameterised variants, structural alterations, alternative data representations, or hybrid approaches. These can be produced manually, via search, or with the assistance of modern generative models.
- 4) Evaluate candidates across the workloads: measure time, memory, stability, robustness, and secondary effects such as cache behaviour.
- 5) Iterate: refine promising candidates, discard unpromising ones, and explore neighbourhoods of the search space suggested by results.

This is in essence the “bare minimum full-sentence” prototype of the idea. Yet even in this minimal form, it conveys a useful principle: one need not commit to a single brilliant insight. Instead, one can explore broadly and let empirical evidence point towards promising refinements.

A. Cascading Opportunities: Ex-Ante and Ex-Post

A critical dimension of this approach concerns not merely the immediate gains from a particular optimization, but the *cascade of opportunities* it may unlock. This evaluation should happen at two distinct moments:

- 1) **Ex-ante assessment (before experimentation):** When considering whether to systematically explore a particular algorithmic target, one should ask: *If this opportunity worked, which other opportunities would it unlock?* This question serves as a multiplier on expected value. An improvement to a widely-used primitive (such as sorting, hashing, or matrix operations) has cascading effects: better primitives enable better compound algorithms, which in turn enable new architectural choices. A representation

change that simplifies one layer of a stack may reveal simplifications in adjacent layers. Thus, the decision to invest in screening should weigh not only the direct performance gain but also the breadth of downstream opportunities.

- 2) **Ex-post learning (after successful results):** Once a successful variant is identified, the analysis deepens: *What characteristics of this particular solution enabled its success, and how can those characteristics be exploited further?* If a new data layout improved cache behavior, which other algorithms share similar access patterns and could benefit? If a hybrid approach outperformed pure methods, what does that reveal about the structure of the workload, and where else might hybridization apply? Success is not merely a better algorithm; it is a probe into the performance landscape, revealing features that guide subsequent exploration.

This dual perspective—anticipating cascades before investment, and extracting generalizable lessons after success—transforms algorithmic screening from isolated optimization into a cumulative, self-reinforcing process. Each successful screening campaign becomes a stepping stone for the next, and the methodology accumulates knowledge about what works and why.

III. WHY THIS COULD BE VALUABLE

The optimism here is not the Californian variety of boundless exuberance. It is the European, academically grounded kind: measured, empirical, aware of what has come before, but open to the possibility that careful experimentation may reveal improvements.

A few quiet observations support this optimism:

- Many algorithms were designed for machine models that no longer exist.
- The performance landscape has evolved: cache hierarchies, vector units, branch predictors, network topologies, software-defined networking, and an ever-growing list of architectural features create opportunities that older analyses did not consider.
- Large-scale automated search has repeatedly discovered structures (in ML, compilers, and optimisation) that humans had overlooked.
- Even small algorithmic improvements, multiplied across the enormous volume of deployed software, can be economically significant.
- A systematic process provides a way to "re-open" established areas without discarding accumulated knowledge.

This is not a promise of dramatic breakthroughs. It is, instead, an invitation to disciplined curiosity. If we accept that the space of possible algorithms is vast and that our current selections reflect historical contingency as much as inherent optimality, then systematic exploration is not only rational but overdue.

A. Scope: Low-Level to Conceptual

The approach applies across multiple levels of abstraction. The examples cited earlier—Gaussian Splatting and attention mechanisms—represent relatively low-level algorithmic changes (a new 3D primitive representation, a new sequence-processing operation) that turned out to have profound high-level impacts across entire domains. Yet the same systematic exploration applies equally to narrower, more localized improvements: optimising the inner loops of a Fourier transform, refining a hash function, or tuning a cache-aware data layout.

What unifies these cases is not the scope of impact, but the method: empirical, structured, and automated exploration of algorithmic variants. In some instances, low-level refinements yield modest but measurable gains. In others, a seemingly local change propagates upward, reshaping how entire systems are built. Both outcomes are valuable, and both are instances of the same underlying concept. Automated exploration is feasible—and potentially fruitful—at every level.

IV. ILLUSTRATIVE EXAMPLE: FFTW

A concrete example helps ground the abstract idea. Consider the canonical problem of the discrete Fourier transform (DFT): despite its long pedigree and extensive study, real-world implementations continue to be refined. The well-known paper "The Fastest Fourier Transform in the West" by Frigo and Johnson (1997) demonstrates this point clearly [3]. The authors built their library (FFTW) by treating the DFT problem as open to systematic improvement:

- They generated *codelets* for small sizes, optimised by a generator, rather than relying only on classic textbook routines.
- At runtime they used a *planner* to choose the composition of these codelets, adapting to the specific hardware (cache sizes, registers, pipelines) instead of using a one-size-fits-all algorithm.
- They achieved significantly better performance than many existing hand-tuned libraries, showing that even a "mature" algorithm domain retained room for innovation.

In other words: an algorithm component previously viewed as a solved area was still amenable to systematic search and optimisation once one applied modern tools and empirical methods.

For our purposes this example illustrates that:

- algorithmic screening is not just for obscure new methods—it applies to common, foundational routines;
- workload characteristics (transform size distributions, memory hierarchies) matter significantly;
- adaptivity and configurability (planner + codelets) mattered more than just a fixed optimal algorithm.

Thus the FFTW case provides both inspiration and validation for the algorithmic screening concept.

V. OUTLOOK

The idea of algorithmic screening is simple, perhaps deceptively so. It requires no philosophical departures, only the

recognition that algorithmic design can benefit from the same empirical, exploratory mindset that has served other scientific disciplines well.

There is no guarantee of dramatic breakthroughs. Yet there is genuine reason to believe that systematic exploration, supported by modern computational tooling, can uncover refinements that matter — and in the process shed new light on how long-established algorithms behave under contemporary conditions.

In this sense, the concept is indeed promising. Quietly so, perhaps, but with a seriousness that justifies attention.

REFERENCES

- [1] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3D Gaussian Splatting for Real-Time Radiance Field Rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, pp. 1–14, July 2023.
- [2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is All You Need," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017, pp. 5998–6008.
- [3] M. Frigo and S. G. Johnson, "The Fastest Fourier Transform in the West," *MIT Lab. for Computer Science, Tech. Report 728*, September 1997.