A Prime Application

by M. PanXnubis A. Sh. Gaia Ladrieh
9/26/25

The Riemann problem, in regards to primes, presents nontrivial zeros (ρ of ζ(s)) to control fluctuations of prime distribution. Zeros (ρ) are effectively cancellation points of interacting waves. Knowing the structure yields precise control of prime counts π(x).

But what if we consructively model the same wave/zero cancellation locally and use geometric/recursion-aware transforms to collapse search space so primes are located exactly and cheaply in practice?

Thesis:

Represent search *S* (candidate numbers in an interval) as geometric constraint space. Use layered constraints (wheel/lattices, small-sieve, matched-filter on CRT residues, toroidal inversions, and algebraic transforms) to collapse the set from exponential to increasingly miniscule. Then verify the remaining candidate with MR/ECPP. This is a semantics to syntax collapse: geometry encodes the semantics so the syntactic verification becomes trivial.

Notation

- *N*: base (e.g., right-hand end of interval)
- Interval [*A,B],* or window *W* = (*N, N* + *B*)
- *W* (sub wheel) = primorial used as wheel (e.g. 2310 = 2 x 3 x 5 x 7 x 11)
- Residues *R* (sub *W*) = {*r* ∈ {1, …, *W*} : gcd(*r,W*) = 1}
- Minimal candidate for residue *r* in window: *n* (sub *r*) = min{*n* ∈ *W : n* ≡ *r* (mod *W*)}
- Trial-div bound *T* (cheap small-factor check)
- MR call = Miller-Rabin invocation on a candidate (each call expensive)
- pop (*r*) = number of popped composite checks observed in residue *r* during a baseline sweep
- Kernel *K(r)* = matched-filter weight for residue *r*
- Score *S(r)* = pop(*r*) ì *K*(*r)*
- Soft-exclude: advancing minimal candidate *n* (sub *r*) ∫ *W* (deprioritie, not delete)
- *P(x)* = Π (sub *I*) (1 + *x*^*w* (sub *I*): generating polynomial for subset-sum (when used)

Wheel/residue lattice (syntax reduction)

Remove trivial residues:

R (sub *W*) {*r* : 1 ≤ *r* ≤ *W*, gcd (*r,*W) = 1}

Candidate set becomes:

C = ∪ (sub *r* ∈ *R* (sub *W*)) {*n* (sub *r*) + *kW* | *n* (sub *r*) + *kW* ≤ *B*} – This immediately removes all

multiples of small primes.

Baseline heap search

Seed a min-heap H with each minimal candidate (n (sub r), r). Repeatedly:

- pop (n, r),
- do trial-division up to T; if composite, record pop for residue r, push $n + W$;
- else do MR (if MR says prime → record prime; else composite → push $n + W$). This produces pop (r) counts and list

Torus matched-filter kernal

Pick a path on $\mathbb{Z}/W$ from an anchor residue to another (e.g., residue of a known prime to residue of N). For each residue r, define:

$K(r) = \Sigma$ (sub p (sub j) ∈ path) exp(- $d(r,p$ (sub j))^2/2σ^2), $d(a,b) = \min(|a - b|, W - |a - b|)$.

The Score:

$S(r) = \text{pop}(r)$ ì $K(r)$

Residues that both produce many pops and sit near the path get prioritized – these are the CRT classes that generate the most costly recursion/entropy.

Soft-exclusion

For top K residues by $S(r)$: test minimal candidate n(sub r) (trial then MR). If composite, *soft-exclude* by advancing minimal candidate by $+W$. This eliminates the costly repeated popping at that residue and dramatically reduces total expensive tests.

Möbius/toroidal inversion and Higher Dimensional transforms

If you embed the candidate indices on a torus (CRT lattice) and perform a Möbius-type inversion (or toroidal degeneration), you can reparametrize the search so large swathes of candidates map to compact polynomial/generating-function objects.

- Generating polynomials: $P(x) = \Pi$ (sub i)$(1 + x^w)$ truncated to degree B.
- Bitset DP: represent reachable sums as bitset $B[s]$ with updates $B \leftarrow B \mid (B << w)$.
- Subset convolution/ SOS DP/NTT: combine buckets efficiently.

Algebraic transforms collapse exponential subset combinatorics into near-linear transforms when window width B is moderate.

Step-by-Step Algorithm

Input: interval $[A,B]$ or window $W = [N, N + B]$; wheel W; trial bound T; small-sieve bound S (sub 1); kernal; K (sub *top*); partition width P (sub *size*); overlap O.

1. Partition [*A,B*] → subintervals *I* (sub 1), …, *I* (sub *m)* with overlaps *O*
2. Per-partition pre-sieve: run segmented sieve of primes ≤ *S* (sub 1) on *I* (sub *j*). Mark trivial composites.
3. Build residues: compute *R* (sub *w*) and minimal *n (*sub *r*) inside *I* (sub *j*).
4. Baseline pass (collect pops):
   -Seed heap *H* with (*n* (sub *r),r*). While *H* not empty: pop candidate, trial-div up to *T*. If composite: record pop (*r*)++, push *n* + *W*. If passes trial-div: do MR; if MR composite record pop; else record prime.
   - Save popped list and pop(*r)*
5. Compute kernel and score: choose path residues (e.g., residues of anchors) and compute *K(r)* and *S(r)* = pop(*r*) ì *K*(*r*).
6. Pretest top residues: for top *K* (sub top) residues by *S(r)*, test only minimal candidate *n* (sub *r*) trial then MR). If composite → soft-exclude *n*(sub *r*) ∫ *n* (sub *r*) + *W* and log (Do *not* delete untested candidates).
7. Iterative search + soft-exclusion: resume heap search with the new minimal pointers – now many pops collapse. Repeat kernel recompute if desired and iterate until heap exhausts.
8. (Optional) Algebraic collapse: when a partition still has dense residue interactions, form generating polynomials per bucket, combine by FFT/NTT/truncated convolution or bitset DP to eliminate entire classes of impossibilities in bulk. This is the subset-sum/DAG move: memoize intermediate polynomials to avoid duplication.
9. Verification: every reported prime must be MR-checked (and optionally ECPP for certs). Merge overlapping partitions and deduplicate. Re-run a final cheap check across overlaps.
10. Output: list of primes + full logs (pops, soft-excludes, MR call count, trial-div calls).

Correctness guarantees

- A candidate is never deleted without verifying it. Soft-exclusion only moves a candidate forward by *W*; it is still in the heap and will be tested later
- Final MR or deterministic certificate required for any reported prime
- Overlapping partitions remove boundary risk. So the algorithm is exact and sound: every reported prime is actually prime; every prime in the interval will be tested in finite time.

Conceptual Mapping

- RH and the explicit formulas show that primes are where a large set of oscillatory terms fail to cancel. Those oscillary terms are the *same* phenomenon as the CRT/lattice superpositions we observe in the heap and pop statistics.
- The matched-filter/torus kernel explicitly looks for residues whose pattern of composite pops signals coherent destructive interference (i.e., where many waves combine to create a heavy composite field). Targeting those residues first is the same as isolating the dominant spectral contributors locally.
- The generating-function (polynomial) view is the finite-window analogue of the zeta's product and its spectral expansion: Π(1-*x*^*p*) vs. Π (1 + *x*^*w*(sub *I*)). Doing truncated, exact convolutions is like computing a localized partial zeta that encodes local superposition behavior – but computationally and constructively.
- Instead of trying to globally characterize all zeros ρ, build a local spectral model (the kernel + algebraic transforms) that expose cancellation structure where it impacts computation. That

allows for the computation of primes exactly without needing global spectral knowledge.

Complexity and Practical Metrics

- Formal worst-case complexity remains governed by prime-gap growth and the number of residues; I am not claiming a universal asymptotic proof that MR calls are polynomial in *log N*.
- Practical metric that matters: number of MR calls and total pops (hea iterations).
- Gains arise because the kernel isolates high-entropy CRT classes; algebraic transforms combine many combinatorial states at once (bitset DP/FFT/NTT), turning exponential branches into polynomial directed transforms when window size is manageable.

Summary

- Riemann's core issue = where waves cancel (zeros) and that controls prime fluctuations.
- My method = building a local spectral model (kernel + algebraic transforms) on the CRT/wheel lattice, finding residues that create maximal entropy, soft-exclude their minima, and use generating-function/DAG memoization to collapse large combinatorial spaces into tractable transforms.
- Result = exact ennumeration/next-prime search with dramatically fewer expensive MR checks in practice; exactness maintained by final primality certificates.

Insights

- Primes as “Next Available Position” in a Constraint Lattice
  - In the wheel/CRT representation, compsites occupy entire resiude classes module small primes.
  - After sieving out those classes, the prime candidates are precisely the “holes” – the next available integers that are coprime to the wheel base.
  - My method shows that primes are not scattered arbitrarily; they are exactly where no smaller modular constraint forbids occupation.
  - This aligns with the viewpoint that primes “take the next available slot” in the lattice, which is why we can predict candidate density very accurately before a single MR test.

- Pop Statistics encode Local Prime Scarcity
  - By counting heap pops per residue class, we are effectively measuring how frequently each CRT class fails.
  - High-pop residues behave like “hot zones” of composite density – they correlate with areas where local prime gaps are slightly larger.
  - Kernel-weighted scoring highlights systematic, not random, structure: residues near a known prime or along a toroidal trajectory show suppressed pops (more likely to host the next prime).

- Local Spectral Model predicts Workload
  - Our torus kernel + matched filter acts like a local approximation to the osciallatory term in the explicit formula for $\psi(x)$.
  - This means we can predict which candidates will be composite-heavy before testing them, reducing work by skipping expensive MR calls on unlikely residues.

- Partition + Memoization collapses Exponential Branching
  - When representing candidates as subset sums of wheel increments, many branches overlap (shared partial sums).
  - Memoization/DAG traversal ensures each unique partial sum is only computed once, effectively performing deterministic dynamic programming over the candidate space.
  - This gives a concrete mechanism for this method scales better: we avoid recomputing the same partial cancellation information over and again.

- Exactness maintained, not just Asymptotic
  - Unlike analytic prime-counting formulas (which are asymptotic), our pipeline produces a guaranteed complete list of primes in the interval.
  - Every candidate is either verified prime (via MR/ECPP) or explicitly shown composite (via trial division or sieve residue exclusion).

- Prime Distriutiong appears Locally Balanced
  - Once conditioned coprime to the wheel base, primes appear with near-constant local density, and pops cluster where the density dips slightly.
  - This confirms that primes fill “available space” almost evenly, which is why a partitioned approach can saely shrink solution space from both ends without missing any primes.

- Operational Metric: MR Call Reduction
  - The most practical indicator of efficiency is number of MR calls per prime found

- Reframing the Prime Problem as a Resource Allocation Problem
  - This entire approach treats primes as outcomes of contraint satisfaction process:
    - The “resource” is available coprime positions.
    - The “allocation is done by systematically eliminating forbidden composites.

  - This reframing allows us to use search-space collapse techniques common in combinatorial optimization (subset convolution, DP, pruning heuristics), but now applied to number theory.