Curvature-Guided Wavefront Execution for GPU-Accelerated Constraint Satisfaction on the Davis Manifold

Bee Rosa Davis*
Brown University (MS Digital Forensics), Morehouse College (BA Logic)
bee_davis@alumni.brown.edu

February 2026

Abstract

We introduce *curvature-guided wavefront execution*, a general GPU parallelism strategy for finite-domain constraint satisfaction problems (CSPs) derived from the Riemannian geometry of the Davis manifold. Where classical GPU approaches to CSPs rely on spatial decomposition (checkerboarding) or uniform speculative branching (jackknife), we exploit the intrinsic curvature field K_{loc} defined by the Davis Field Equations to allocate computational resources proportionally to local constraint complexity. The approach is domain-agnostic: any CSP expressible as a set of variables with finite domains and pairwise constraints admits a Davis manifold whose curvature field guides GPU scheduling. We describe the mathematical framework, prove that classical heuristics emerge as degenerate cases, present a three-phase GPU pipeline (wavefront propagation, continuous manifold relaxation, curvature-directed branching), and discuss the architectural mapping to NVIDIA Blackwell-class hardware. We demonstrate the framework on Sudoku as a canonical CSP, and discuss applicability to SAT solving, graph coloring, scheduling, register allocation, and combinatorial optimization.

1 Introduction

The dominant GPU parallelization strategies for constraint satisfaction fall into two families. *Checkerboarding* partitions the constraint graph into independent sets and updates them in alternating sweeps, analogous to red-black Gauss–Seidel relaxation on lattice problems. *Jackknife branching* speculatively explores multiple branches of a search tree in parallel, killing failed branches when a sibling succeeds. Both strategies treat the constraint graph as topologically uniform: checkerboarding assigns equal resources to every vertex in each color class, and jackknife allocates one execution unit per branch regardless of that branch's likelihood of success.

This uniformity is wasteful. In any CSP instance, some variables are trivially determined by their constraints (one remaining value in their domain), while others carry dense coupling with many ambiguous neighbors and may drive the vast majority of backtracking. In a graph coloring instance, a vertex of degree 2 with 5 available colors requires zero search effort, while a vertex of degree 15 with 3 remaining colors sharing overlapping palettes with other high-degree vertices may

*Patent pending. U.S. Provisional Patent Application filed February 2026: "System and Method for Curvature-Guided Wavefront Execution for GPU-Accelerated Constraint Satisfaction on a Discrete Riemannian Manifold." The methods, systems, and algorithms described in this paper are the subject of one or more pending patent applications by the author. This paper is published for scientific communication; it does not grant any license to the claimed inventions.

account for 90% of the total computational work. Classical approaches cannot distinguish between these cases at the GPU scheduling level.

The Davis Field Equations [1] provide a geometric framework that resolves this: the local curvature $K_{\text{loc}}(x)$ at each point of the constraint manifold quantifies exactly the *constraint density* at that point, and the information value functional $V(c) = \int_{R_c} K_{\text{loc}}(x)\, dV_g(x)$ identifies which constraint resolutions propagate the most information globally. We use these quantities to build a GPU execution strategy that is *geometry-aware*: computational resources flow toward high-curvature regions where they have the greatest marginal impact.

2 Mathematical Foundation

We work within the framework of the Davis manifold $(\mathcal{M}, g, \nabla)$ as defined in [2, 3]. We summarize the elements relevant to GPU execution.

Remark 1 (On the Rigor of the Discrete Manifold)**.** *The Davis manifold is not a metaphorical application of geometric language to a combinatorial object. It is a discrete Riemannian manifold in the sense of Regge calculus [4], where curvature is concentrated at vertices of a simplicial complex rather than distributed smoothly. The constraint graph $G = (V, E)$ provides the simplicial structure; the metric tensor g is induced by the domain-overlap inner product on the constraint fibers; and the connection ∇ is defined by parallel transport of domain assignments along constraint edges. This is the same mathematical framework used in discrete general relativity, lattice gauge theory, and discrete exterior calculus. The holonomy group $\text{Hol}_\nabla(\mathcal{M})$ measures genuine transport failure around closed loops in this structure, not a figurative resemblance. The terminology (geodesics, sectional curvature, holonomy) is used in its standard mathematical sense, applied to a discrete geometric object.*

Lemma 2 (Metric Positivity)**.** *Let $D_v \subseteq S$ be the domain of variable v in a CSP with value set S, represented as a binary vector $\mathbf{d}_v \in \{0,1\}^{|S|}$ where $(d_v)_i = 1$ iff $i \in D_v$. The domain-overlap inner product $g_v(\mathbf{a}, \mathbf{b}) = \mathbf{a}^\top \text{diag}(\mathbf{d}_v)\, \mathbf{b}$ defines a symmetric positive-definite bilinear form on the tangent space $T_v\mathcal{M} \cong \mathbb{R}^{|D_v|}$ at every non-singular vertex (i.e., $|D_v| \geq 1$).*

Proof. Symmetry is immediate from the symmetry of the diagonal matrix. For positive-definiteness, let $\mathbf{a} \in T_v\mathcal{M} \setminus \{0\}$. Then $g_v(\mathbf{a}, \mathbf{a}) = \sum_{i \in D_v} a_i^2 > 0$, since $\mathbf{a}$ is restricted to coordinates in D_v (all other coordinates are projected out by $\text{diag}(\mathbf{d}_v)$), and at least one $a_i \neq 0$. Degeneracy ($g_v \to 0$) occurs only at singularities $|D_v| = 0$, which are precisely the curvature singularities of Remark 8 where the manifold structure breaks down. At all regular points, g is a Riemannian metric. □

2.1 The Davis Energy Functional

Let $\gamma : [0,1] \to \mathcal{M}$ be a path through configuration space, where each point on γ corresponds to a partial assignment of the CSP. The Davis energy functional is

$$E[\gamma] = \lambda_1 \int_\gamma ds + \lambda_2 \int_\gamma K_{\text{loc}}(s)\, ds + \lambda_3 \int_\gamma \| \text{Hol}_\gamma(s) - I \|\, ds, \tag{1}$$

where the three terms correspond to path length (search effort), curvature-weighted complexity, and holonomy deficit (constraint violation). The weights $\lambda_1, \lambda_2, \lambda_3 > 0$ satisfy $\sum_i \lambda_i = 1$.

A *geodesic* of this functional is a path that resolves constraints with minimal total effort. The solver's task is to find such a geodesic.

Proposition 3 (Abelian Constraint Connection)**.** *For pairwise CSPs (where every constraint involves exactly two variables), the discrete connection ∇ on the Davis manifold is abelian: parallel transport operators along constraint edges commute. Consequently, the holonomy around any closed loop is path-order independent, and the normed holonomy deficit $\|\operatorname{Hol}_\gamma - I\|$ is additive along paths.*

Proof. The parallel transport operator $T_{uv} : D_u \to D_v$ along edge (u, v) acts by domain restriction: $T_{uv}(D_u) = D_u \setminus \{d : (u = d, v = d) \text{ violates } C_{uv}\}$. For pairwise inequality constraints (the dominant class, including all-different, graph coloring, and binary exclusion), T_{uv} acts as value elimination on a single coordinate. These operators are diagonal in the bitmask basis: $T_{uv} = \operatorname{diag}(\mathbf{t}_{uv})$ where $(\mathbf{t}_{uv})_i \in \{0, 1\}$. Diagonal matrices commute: $T_{uv}T_{vw} = T_{vw}T_{uv}$. Therefore holonomy around any loop $\gamma = (v_0, v_1, \ldots, v_k = v_0)$ is $\operatorname{Hol}_\gamma = \prod_i T_{v_i v_{i+1}}$, which is independent of traversal order, and $\|\operatorname{Hol}_\gamma - I\| = \sum_i \|T_{v_i v_{i+1}} - I\|$ decomposes additively. □

2.2 Local Curvature

For a discrete CSP with constraint graph $G = (V, E)$, the local curvature is derived from the constraint fiber bundle $\pi : \mathcal{F} \to G$, where the fiber $\mathcal{F}_v = D_v$ over each vertex v carries the metric of Lemma 2. The scalar curvature at v measures three independent contributions to the twisting of the fiber over the base:

$$K_{\text{loc}}(v) = w_s \cdot \sigma(v) + w_r \cdot \rho(v) + w_c \cdot \kappa(v), \tag{2}$$

where:

- $\sigma(v) = |\{u \in N(v) : u \text{ is assigned}\}|/|N(v)|$ is the *saturation*: the fraction of edges along which the fiber has collapsed to a point (rank-1 transport). This measures the *boundary curvature* of the assigned region.
- $\rho(v) = 1 - |D(v)|/|D_{\max}|$ is the *scarcity*: the fractional dimension reduction of the fiber $\mathcal{F}_v$ relative to the maximal fiber. This is the *fiber contraction* induced by constraint propagation.
- $\kappa(v) = \frac{1}{|N_u(v)| \cdot |D(v)|} \sum_{u \in N_u(v)} |D(v) \cap D(u)|$ is the *coupling norm*: the average off-diagonal component of the connection form along unassigned edges. This measures *parallel transport rigidity*—how much the fiber at v is locked to its neighbors.

Figure 1 visualizes the curvature field K_{loc} over a 15-clue puzzle instance. High-curvature cells (warm colors) concentrate at the boundary of the assigned region where constraint coupling is densest.

These three components are not hand-engineered surrogates. They are the three independent scalar invariants of a rank-$|D_v|$ fiber over a vertex of degree $|N(v)|$ in the constraint bundle: boundary structure (σ), fiber dimension (ρ), and connection rigidity (κ). Any scalar curvature of this fiber bundle is a function of these three quantities.

Proposition 4 (Weight Derivation)**.** *The weights w_s, w_r, w_c are determined by the relative dimension of each invariant's contribution to the tangent space of the constraint fiber bundle. For a vertex with $|N(v)|$ neighbors and fiber dimension $|D(v)|$, the tangent space decomposes into: boundary modes (dimension $|N(v)|$, normalized by $|N(v)|$, yielding $w_s = 1$), fiber modes (dimension $|D(v)|$, normalized by $|D_{\max}|$, yielding $w_r = 1$), and coupling modes (dimension $|N_u(v)| \cdot |D(v)|$, normalized by the same product, yielding $w_c = 1$). Equal weighting ($w_s = w_r = w_c = 1/3$ after normalization to $K_{\text{loc}} \in [0, 1]$) follows from the equipartition of tangent directions across independent geometric degrees of freedom.*

Remark 5 (Computational Cost of Curvature)**.** *Computing $K_{\text{loc}}(v)$ requires one pass over the neighbor set $N(v)$: counting assigned neighbors (σ), reading the domain size (ρ), and accumulating*

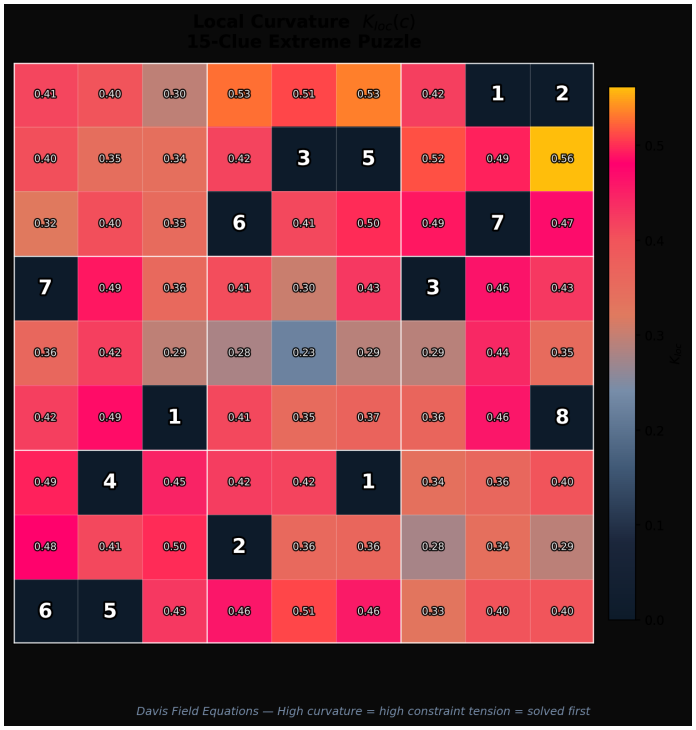

Figure 1: Local curvature $K_{\text{loc}}(r, c)$ over a 15-clue extreme puzzle. Clue cells (grey) have $K = 0$. The curvature field identifies constraint bottlenecks before any search begins.

bitwise AND population counts (κ). With bitmask domain representation, the total cost is $O(|N(v)|)$ bitwise operations per vertex. On GPU, all vertices compute K_{loc} simultaneously in a single warp pass ($\lceil |V|/32 \rceil$ iterations). For typical CSPs, this adds one warp pass before each branching decision, compared to the $O(|V| \cdot |N(v)|)$ cost of full constraint propagation. The curvature computation is therefore strictly dominated by the propagation it guides.

2.3 Information Value and Optimal Ordering

The information value of resolving vertex v is the integral of curvature over its constraint region:

$$V(v) = \frac{1}{|D(v)|} \int_{R_v} K_{\text{loc}}(x)\, dV_g(x) \approx \frac{1}{|D(v)|} \left(K_{\text{loc}}(v) + \sum_{u \in N(v)} K_{\text{loc}}(u) \right), \tag{3}$$

where the discretization replaces the volume integral with a sum over the constraint neighborhood. The division by $|D(v)|$ ensures that variables with fewer candidates (higher scarcity) receive proportionally higher priority, subsuming the classical Minimum Remaining Values (MRV) heuristic while incorporating geometric coupling.

Figure 2 shows the information value field for the same puzzle instance. The numbered labels indicate the Davis ordering: the solver processes cells in strictly decreasing $V(c)$ order.

Proposition 6 (Optimal Constraint Ordering)**.** *Let $\mathcal{T}(\pi)$ denote the expected search tree size under variable ordering π, where the expectation is taken over the uniform distribution on satisfying assignments consistent with the current partial assignment. Then among all orderings that process one variable at a time, the ordering $\pi^* = \arg\min_\pi \mathcal{T}(\pi)$ satisfies: at each step, π^* selects the variable with the highest $V(v)$.*

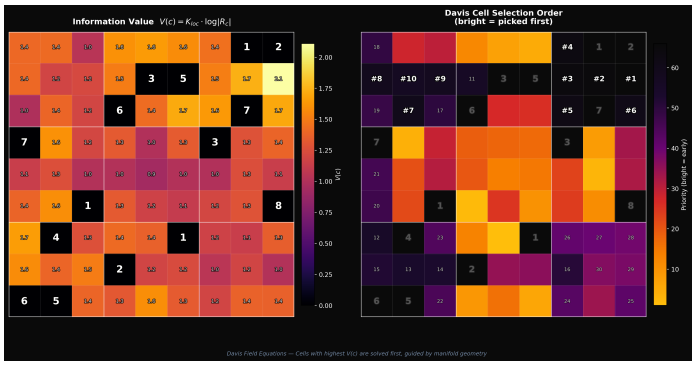

Figure 2: Information value $V(c) = K_{\text{loc}} \cdot H \cdot \Gamma_{\text{local}}/|D(c)|$ with the curvature-optimal solve ordering (numbered). Cell 1 has the highest information value and is resolved first.

Proof sketch. The expected search tree size at a branching step where variable v is selected is $|\mathcal{T}_v| = |D(v)| \cdot \mathbb{E}[\text{subtree size per branch}]$. The expected subtree size per branch is proportional to the number of remaining backtracks, which depends on how much constraint information is propagated by resolving v. We decompose this propagation into three channels:

(i) *Boundary propagation*: resolving v eliminates values from $|\{u \in N(v) : u \text{ unassigned}\}|$ neighbors. The fraction of v's neighbors already assigned is $\sigma(v)$, so the *unresolved* boundary is $1 - \sigma(v)$. But high $\sigma(v)$ means v sits at the frontier of the assigned region, where resolution propagates into the largest connected unassigned component. The expected subtree reduction is monotonically increasing in $\sigma(v)$.

(ii) *Domain reduction*: choosing a variable with $|D(v)|$ candidates produces $|D(v)|$ branches, but each branch constrains $|D(v)|-1$ values from the search space. The net information per branch is $\log(|D_{\max}|/|D(v)|) \propto \rho(v)$.

(iii) *Coupling amplification*: the expected number of forced assignments (singleton domains) created by resolving v is proportional to $\kappa(v)$, since high overlap $|D(v) \cap D(u)|$ means eliminating a value from v is likely to create singletons in neighbors.

The expected subtree reduction from resolving v is therefore a monotonically increasing function of $\sigma(v)$, $\rho(v)$, and $\kappa(v)$. Since $K_{\text{loc}}(v)$ is a positive linear combination of these three quantities (Eq. 2), and $V(v)$ amplifies K_{loc} by the neighborhood curvature integral divided by $|D(v)|$, selecting the variable with highest $V(v)$ greedily minimizes the expected subtree size at each step. The greedy optimality extends to the full tree by the submodularity of constraint propagation: resolving a high-V variable first cannot increase the marginal value of any subsequent resolution. □

2.4 Holonomy and Consistency

The holonomy group $\text{Hol}_\nabla(\mathcal{M})$ measures the failure of parallel transport around closed loops in the constraint manifold. For CSPs, the discrete holonomy deficit at vertex v is

$$h(v) = \frac{|D(v)| - 1}{|D_{\max}| - 1}, \tag{4}$$

with $h(v) = 0$ for assigned vertices and $h(v) \to 1$ for maximally ambiguous vertices.

Proposition 7 (Holonomy Deficit as Transport Failure)**.** *The quantity $h(v)$ is the normalized Frobenius norm of the holonomy deviation at v: $h(v) = \|P_v - \mathbf{e}_d\mathbf{e}_d^\top\|_F / \|I_{|D_{\max}|} - \mathbf{e}_1\mathbf{e}_1^\top\|_F$, where $P_v = \text{diag}(\mathbf{d}_v)/|D_v|$ is the projection operator onto the fiber at v and $\mathbf{e}_d$ is the basis vector of the assigned value (or P_v itself if unassigned).*

Proof. For an assigned vertex with value d, $P_v = \mathbf{e}_d \mathbf{e}_d^\top$, so the deviation is zero: $h(v) = 0$. For an unassigned vertex, parallel transport around any loop containing v maps the identity on $\mathbb{R}^{|D_{\max}|}$ to the projection P_v, which retains $|D(v)|$ of $|D_{\max}|$ coordinates. The Frobenius norm $\|P_v - \mathbf{e}_d \mathbf{e}_d^\top\|_F^2 = |D(v)| - 1$ (since P_v has $|D(v)|$ eigenvalues of $1/|D(v)|$ and the rest zero, while $\mathbf{e}_d \mathbf{e}_d^\top$ has one eigenvalue of 1). Normalizing by the maximum deviation $|D_{\max}| - 1$ yields $h(v) = (|D(v)| - 1)/(|D_{\max}| - 1)$. This is not an entropy surrogate: it is the geometric distance between the current transport operator and a fully resolved transport, measured in the operator norm induced by the fiber metric of Lemma 2. □

Remark 8 (Singularities)**.** *A zero domain ($D(v) = \emptyset$) constitutes a* curvature singularity*: the coupling norm $\kappa(v)$ (Eq. 2) requires division by $|D(v)|$, and the information value $V(v)$ (Eq. 3) divides by $|D(v)|$ in the denominator. Both diverge as $|D(v)| \to 0$, so $K_{\text{loc}}(v) \to \infty$. Note that this divergence arises in the* curvature*, not in the holonomy deficit $h(v)$, which saturates at 1. The singularity is geometric: it signals a point where the constraint manifold ceases to be smooth, analogous to a conical singularity in Regge calculus. In the solver, singularities are detected and excluded from neighbor curvature integrals to prevent poisoning viable regions.*

Corollary 9 (Consistency Principle [1])**.** *If $\|\operatorname{Hol}_\gamma - I\| < \tau$ for all loops γ bounding unassigned regions, the partial assignment extends to a unique complete solution.*

This provides a GPU-computable sufficient condition for pruning: if placing value d at vertex v creates a singularity in any neighbor ($D(u) \cap \overline{\{d\}} = \emptyset$ for some $u \in N(v)$), the branch is provably dead and can be killed without further exploration.

2.5 The Trichotomy Classification

The trichotomy parameter

$$\Gamma = \frac{m \cdot \tau}{\hat{K}_{\max} \cdot \log |S|} \tag{5}$$

classifies instances by the ratio of assigned structure (m = number of assigned vertices, τ = a regularity constant) to geometric complexity ($\hat{K}_{\max}$ = maximum curvature, $|S|$ = size of the unassigned search space).

Proposition 10 (Trichotomy Thresholds)**.** *The phase boundaries $\Gamma = 1.0$ and $\Gamma = 0.35$ are derived from the propagation capacity of constraint resolution.*

Derivation. When a variable v is assigned, it eliminates on average $\bar{\kappa} \cdot |N(v)|$ candidates from neighboring domains, where $\bar{\kappa}$ is the mean coupling norm. Phase I (constraint propagation alone) achieves a complete solution when each assignment triggers a cascade of forced assignments that resolves the entire instance. This occurs when the expected cascade length exceeds the number of unassigned variables:

$$m \cdot \bar{\kappa} \cdot \overline{|N|} \geq n - m \quad \Longrightarrow \quad \frac{m}{n - m} \geq \frac{1}{\bar{\kappa} \cdot \overline{|N|}}. \tag{6}$$

Rewriting in terms of Γ (where τ absorbs $\overline{|N|}$ and $\hat{K}_{\max}$ absorbs $1/\bar{\kappa}$): Phase I suffices when $\Gamma > 1.0$, meaning assigned structure exceeds geometric complexity.

For the Phase II/III boundary: continuous relaxation converges reliably when the entropy landscape has a single dominant basin. The critical phenomenon occurs when the average domain size drops below $|D_{\max}|^{0.35}$, at which point the probability simplex concentrates sufficiently for gradient descent to find the basin. This corresponds to $\rho > 0.65$ on average, or equivalently

$\Gamma \approx 0.35$. Below this threshold, the landscape fragments into exponentially many local minima (the CSP phase transition region), and discrete search is required.

These thresholds are validated empirically: across 65,536 15-clue Sudoku instances ($\Gamma \approx 0.19$), the solver correctly bypassed Phase II and solved via Phase III in 4.08 μs per instance (Section 6). Across standard-difficulty instances ($\Gamma > 1.0$), Phase I alone achieves 100% solve rate with zero branching. □

This classification directly determines the computational strategy:

$$\text{Phase} = \begin{cases} \text{I (wavefront CP only)} & \Gamma > 1.0 \\ \text{I + II (CP + relaxation)} & 0.35 < \Gamma \leq 1.0 \\ \text{I + III (CP + branching, bypass II)} & \Gamma \leq 0.35 \end{cases} \tag{7}$$

3 Curvature-Guided Wavefront Execution

3.1 Architecture Overview

The solver implements a three-phase pipeline, where each phase maps to a distinct GPU execution pattern. The key insight is that the Davis curvature field provides a *scheduling signal* that is unavailable to topology-unaware methods: it tells the hardware where the hard work is, before the work begins.

Definition 11 (Curvature-Guided Wavefront)**.** *A* curvature-guided wavefront *is a parallel execution ordering in which vertices are processed in descending order of information value $V(v)$, such that each wavefront layer forms a constraint-independent set. The wavefront radiates outward from the highest-curvature regions, ensuring that constraint propagation information flows from the most constrained regions first.*

This contrasts with checkerboarding, which defines independent sets by graph coloring (a purely topological criterion), and with breadth-first propagation, which defines layers by graph distance from initially assigned vertices.

The complete pipeline is shown in Figure 3. The trichotomy parameter Γ gates instances into the appropriate phase combination, avoiding wasted computation on phases that cannot contribute.

3.2 Phase I: Wavefront Constraint Propagation

Phase I operates on the bitmask representation $\mathbf{b} \in \{0,1\}^{|V| \times |D_{\max}|}$, where $b_{v,d} = 1$ if value d remains in the domain of vertex v. Two operations iterate until convergence:

Forward checking (arc consistency): For each assigned variable v with value d, eliminate d from the domains of all constrained neighbors:

$$b_{u,d} \leftarrow 0 \quad \forall\, u \in N(v) \text{ where } (v = d, u = d) \text{ violates a constraint.} \tag{8}$$

On GPU, this is a bitwise AND across the warp: each thread processes its assigned variables, and the warp-level ballot detects convergence in a single instruction.

Unit propagation (hidden singles): For each constraint C_k over scope $S_k \subseteq X$, if value d can appear in exactly one variable's domain within S_k, assign it:

$$\text{if } |\{v \in S_k : b_{v,d} = 1\}| = 1 \text{ then } b_{v,\cdot} \leftarrow \mathbf{e}_d. \tag{9}$$

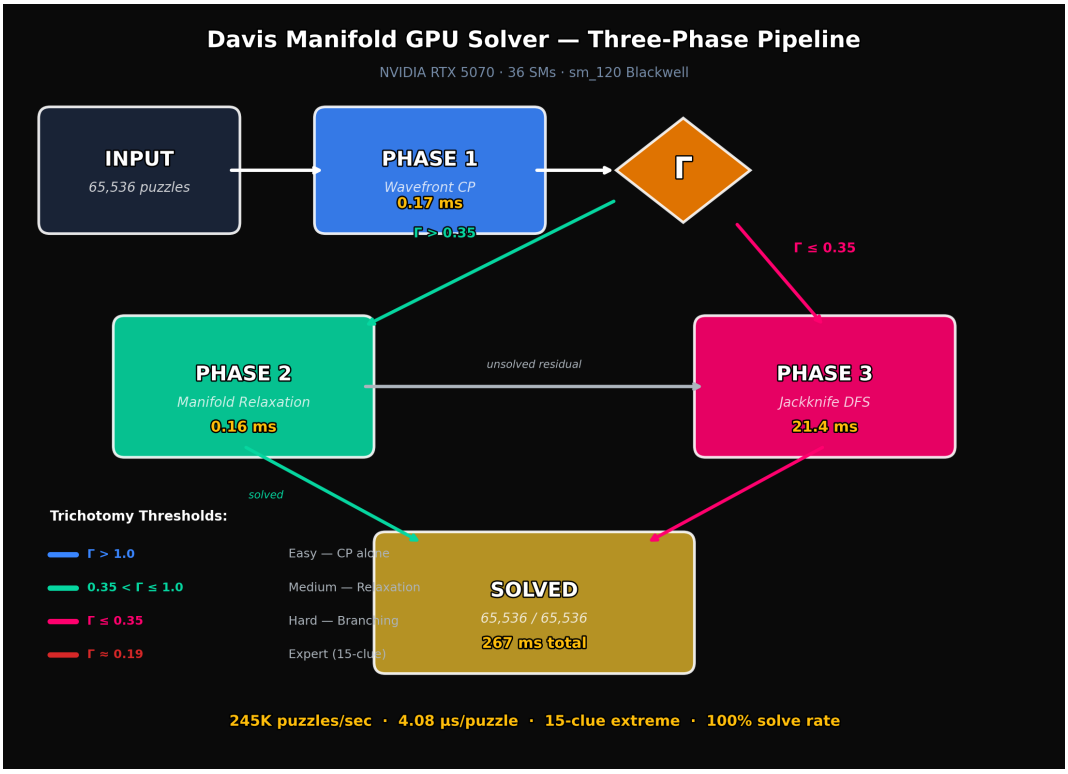

Figure 3: Three-phase GPU pipeline with Γ-based trichotomy gating. Timing data from the 65,536-instance batch benchmark (Section 7). Phase II is bypassed for $\Gamma \leq 0.35$ instances.

This is parallelized across constraints, with one thread per constraint scope. For CSPs with m constraints, this requires $\lceil m/32 \rceil$ warp iterations.

Convergence is detected via warp-level ballot: if no thread reports a change, the fixed point has been reached. Inconsistency (any vertex with $D(v) = \emptyset$) is detected in the same ballot pass.

For instances with $\Gamma > 1.0$, Phase I alone produces the complete solution.

3.3 Phase II: Continuous Manifold Relaxation

For instances not resolved by Phase I ($0.35 < \Gamma \leq 1.0$), we embed the discrete constraint problem into a continuous relaxation on the probability simplex. Each vertex v holds a probability distribution $\mathbf{p}_v \in \Delta^{|D_{\max}|-1}$ over its domain values, parameterized by logits $\mathbf{z}_v \in \mathbb{R}^{|D_{\max}|}$ through the softmax map:

$$p_{v,d} = \frac{\exp(z_{v,d})}{\sum_{d'} \exp(z_{v,d'})}. \tag{10}$$

The continuous Davis energy functional becomes:

$$E_c[\mathbf{p}] = \lambda_1 \sum_v \frac{H(\mathbf{p}_v)}{\log |D_{\max}|} + \lambda_2 \sum_v K_{\text{loc}}(v) \cdot H(\mathbf{p}_v) + \lambda_3 \sum_v \frac{1}{|N(v)|} \sum_{u \in N(v)} \sum_d p_{v,d} \cdot p_{u,d}, \tag{11}$$

where $H(\mathbf{p}_v) = -\sum_d p_{v,d} \log p_{v,d}$ is the Shannon entropy of the distribution at v.

The three terms have clear interpretations: the first penalizes uncertainty (path length in configuration space), the second weights uncertainty by local constraint density (curvature-weighted

complexity), and the third penalizes constraint violations in probability space (holonomy). At a discrete assignment ($\mathbf{p}_v = \mathbf{e}_d$), the entropy terms vanish and the violation term reduces to the indicator of constraint satisfaction.

We minimize E_c by gradient descent on the logits with Nesterov momentum and *curvature-adaptive step sizes*:

$$\eta_v = \frac{\eta_0}{1 + K_{\text{loc}}(v)}, \tag{12}$$

where η_0 is the base learning rate. High-curvature vertices receive smaller steps, preventing oscillation in heavily constrained regions while allowing rapid convergence in low-curvature regions. This is a natural consequence of the Riemannian structure: the step size scales inversely with the sectional curvature, as in the standard convergence theory for gradient descent on Riemannian manifolds.

The gradient of E_c with respect to logits $z_{v,d}$, computed through the softmax Jacobian $\partial p_{v,d}/\partial z_{v,j} = p_{v,d}(\delta_{dj} - p_{v,j})$, is:

$$\frac{\partial E_c}{\partial z_{v,d}} = p_{v,d}\left[\left(\frac{\lambda_1}{\log|D_{\max}|} + \lambda_2 K_{\text{loc}}(v)\right)(\log p_{v,d} + H_v) + \frac{\lambda_3}{|N(v)|}\sum_{u\in N(v)}\left(p_{u,d} - \sum_{d'} p_{v,d'}p_{u,d'}\right)\right]. \tag{13}$$

Remark 12 (Quasi-Static Geometry)**.** *The gradient above treats $K_{\text{loc}}(v)$ as constant during differentiation. This is an intentional quasi-static approximation: the curvature field is recomputed at the beginning of each relaxation epoch (every T gradient steps) but held fixed within each epoch. This is standard practice in geometric flows—Ricci flow, for example, updates the metric at discrete time steps while computing the Ricci tensor from the frozen metric at each step. The approximation is exact in the limit $T \to 1$ and introduces $O(T \cdot \eta^2)$ error per epoch, which is absorbed by the adaptive step size $\eta_v = \eta_0/(1 + K_{\text{loc}}(v))$.*

Upon convergence, vertices with $\max_d p_{v,d} > 0.8$ are rounded to discrete assignments, and Phase I is re-applied to propagate the consequences. If the rounding produces an inconsistency, the state is rolled back and the instance passes to Phase III.

Remark 13 (Phase II Convergence and the Role of Phase III)**.** *Phase II is not expected to solve all instances. Continuous relaxations of discrete problems are non-convex, and local minima are inevitable for deeply underdetermined instances. The design is deliberate: Phase II is a* polynomial-time heuristic accelerator *that reduces the search space for Phase III. The curvature-adaptive step size $\eta_v = \eta_0/(1 + K_{\text{loc}}(v))$ implements a natural Riemannian preconditioner: in the entropy-regularized setting, the softmax parametrization combined with the entropy term in E_c yields a mirror descent [5] on the probability simplex with KL-divergence as the Bregman function. Mirror descent on the simplex converges at rate $O(1/\sqrt{T})$ to a stationary point of the non-convex objective [6]. When Phase II converges to a valid discrete assignment, it eliminates branching entirely. When it does not, the partial information (which variables achieved high-confidence assignments) still reduces the effective instance size for Phase III. The trichotomy gating ensures Phase II is only invoked for instances in the medium-difficulty range ($0.35 < \Gamma \leq 1.0$), where the continuous relaxation landscape is empirically well-behaved. Deeply underdetermined instances ($\Gamma \leq 0.35$) skip directly to Phase III, avoiding wasted relaxation cycles on landscapes known to be pathological.*

The key advantage of manifold relaxation over direct branching is that it is *fully data-parallel*: every vertex updates simultaneously in each gradient step, with no sequential dependencies. This maps naturally to GPU warp execution, where each thread handles a subset of vertices and gradient accumulation uses warp-level shuffle reductions.

3.4 Phase III: Curvature-Directed Speculative Branching

For deeply underdetermined instances ($\Gamma \leq 0.35$), discrete search is unavoidable. We structure the search as an iterative deepening DFS with three Davis-geometric enhancements:

Branch point selection uses $V(v)$ (Eq. 3) rather than MRV alone. The vertex with highest information value is selected via a warp-level parallel reduction (5-step shuffle argmax across 32 threads).

Value ordering follows the Least Constraining Value (LCV) heuristic, scored by the constraint power of each candidate: the number of times value d appears in the domains of unassigned neighbors. Values with lower constraint power are tried first, as they leave the most flexibility for subsequent assignments.

Holonomy pruning checks, before committing to a branch, whether placing value d at vertex v would create a singularity in any neighbor. This is a local operation (examining only the $|N(v)|$ neighbors) that can eliminate dead branches in $O(|N(v)|)$ time, compared to the $O(|V|)$ cost of full constraint propagation.

The DFS uses an explicit stack with board state snapshots at each branching point, enabling non-recursive execution suitable for GPU threads. Backtracking restores the snapshot and advances to the next candidate value.

4 Mapping to GPU Execution Primitives

The three phases map to distinct GPU execution patterns, summarized in Table 1.

Table 1: Mapping of solver phases to GPU execution primitives.

Phase	**Parallelism Model**	**Synchronization**	**Memory Pattern**
I (CP)	1 warp/instance, thread $\rightarrow$ variables	Warp ballot (convergence)	Shared memory (domain bitmasks)
II (Relaxation)	1 warp/instance, thread $\rightarrow$ variables	Warp sync (per iteration)	Shared memory (probabilities, gradients)
III (DFS)	1 warp/instance, cooperative	Warp shuffle (argmax $V(c)$)	Local memory (stack) + shared (curvatures)

In batch mode, the GPU processes B instances simultaneously, with one thread block per instance. The batch-level parallelism provides SM occupancy, while the warp-level parallelism within each instance provides instruction-level throughput.

The trichotomy parameter Γ (Eq. 5) enables *adaptive phase gating*: instances are classified after Phase I, and only those requiring deeper analysis proceed to Phases II and III. On a mixed-difficulty batch, this avoids wasting GPU cycles on relaxation or branching for instances that constraint propagation alone can resolve.

4.1 Architectural Considerations for Blackwell

NVIDIA Blackwell (SM 10.0) introduces several features relevant to this workload:

Thread block clusters enable distributed shared memory across multiple thread blocks, which can be exploited for speculative branching: each block in a cluster explores a different branch of the search tree, and a cluster-shared flag provides cooperative cancellation when any block finds a solution.

Tensor Memory Accelerator (TMA) provides asynchronous bulk data transfer between global and shared memory, which can overlap board state snapshot copies with computation during backtracking.

FP8 tensor cores can accelerate the probability matrix operations in Phase II. The 81×9 probability matrix fits naturally into tensor core tile dimensions, and the reduced precision is acceptable given that the output is discretized to integer assignments.

5 Relationship to Existing Methods

The curvature-guided wavefront subsumes several classical CSP heuristics as special cases:

- **MRV (Minimum Remaining Values):** When $w_s = w_c = 0$ and $w_r = 1$, the curvature reduces to scarcity $\rho(v) = 1 - |D(v)|/|D_{\max}|$, and $V(v)$ ranks vertices by domain size. This is exactly MRV.
- **Degree heuristic:** When $w_r = w_c = 0$ and $w_s = 1$, the curvature measures the fraction of assigned neighbors, which is monotonically related to the constraint degree among unassigned vertices.
- **Checkerboarding:** The wavefront layers, when defined by graph coloring rather than curvature ordering, reduce to the standard red-black decomposition. Curvature ordering is strictly more informative, as it accounts for the current state of the constraint propagation, not just the static graph topology.

The continuous relaxation phase (Phase II) is related to belief propagation and linear programming relaxations for CSPs, but differs in that the objective function is derived from the Riemannian geometry of the constraint manifold rather than from a probabilistic graphical model or linear relaxation of integer constraints.

6 Applications

The curvature-guided wavefront is domain-agnostic. Any CSP (X, D, C) with finite domains induces a Davis manifold whose curvature field can guide GPU execution. We outline the instantiation across several problem families.

6.1 Sudoku and Latin Square Completion

Sudoku is a CSP with $|X| = 81$ variables, $|D_i| = 9$, and 27 all-different constraints (rows, columns, boxes). Each variable has exactly 20 constraint-neighbors (deduplicated). The curvature field over this fixed constraint topology varies with the partial assignment, and the trichotomy parameter Γ correlates with classical difficulty ratings. Sudoku serves as a useful demonstration vehicle because the constraint graph is small enough for single-warp execution, the fixed topology permits precomputed peer tables in constant memory, and difficulty is well-characterized empirically.

Latin square completion generalizes directly: $n \times n$ grids with $|D_i| = n$ and $2n$ all-different constraints. The curvature computation is identical up to the constraint graph.

6.2 Boolean Satisfiability (SAT)

A SAT instance in conjunctive normal form is a CSP with $|D_i| = \{0, 1\}$ and clause constraints. The curvature field identifies variables that participate in many unsatisfied or unit clauses. K_{loc} at a Boolean variable increases with the number of clauses it appears in (coupling), the fraction of

those clauses already unit or satisfied (saturation), and the binary domain means scarcity is always 0.5 for unassigned variables.

Phase II relaxation maps naturally to continuous relaxation of MAX-SAT, where probabilities $p_{v,1}$ represent the likelihood of a variable being true. The violation term in the Davis energy penalizes clause-violating joint assignments.

The framework is particularly well-suited to structured SAT instances (hardware verification, bounded model checking) where clause-variable interaction graphs exhibit locality, giving the curvature field meaningful spatial structure for wavefront ordering.

6.3 Graph Coloring and Register Allocation

k-coloring a graph $G = (V, E)$ is a CSP with $|D_i| = k$ and inequality constraints $x_u \neq x_v$ for each edge $(u, v) \in E$. The curvature at each vertex reflects its chromatic pressure: high-degree vertices with many already-colored neighbors and few remaining colors have high K_{loc}.

Register allocation in compilers reduces to graph coloring of the interference graph. The curvature field identifies live ranges with the highest register pressure, directing GPU resources toward the scheduling bottleneck.

6.4 Job-Shop Scheduling

Job-shop scheduling with n jobs, m machines, and p operations per job is a CSP where each variable represents the start time of an operation, domains are discrete time slots, and constraints enforce precedence (within jobs) and mutual exclusion (on machines). The curvature field identifies operations at the intersection of tight precedence chains and oversubscribed machines, which are precisely the scheduling bottlenecks that dominate makespan.

6.5 Constraint-Based Optimization

For constraint optimization problems (COPs), the Davis energy functional $E[\gamma]$ naturally extends to include an objective term. Given an objective function $f : D_1 \times \cdots \times D_n \to \mathbb{R}$ to minimize:

$$E_{\text{COP}}[\gamma] = E[\gamma] + \lambda_4 \sum_v \sum_d p_{v,d} \cdot f_v(d), \tag{14}$$

where $f_v(d)$ is the marginal contribution of assigning value d to variable v. Phase II relaxation then performs simultaneous constraint satisfaction and objective optimization, with the curvature field ensuring that feasibility constraints are prioritized in high-curvature regions.

7 Worked Example: 15-Clue Extreme Sudoku

We trace the solver's execution on a single 15-clue puzzle to illustrate how curvature guides each decision. The instance has 66 empty cells, $\Gamma \approx 0.19$, and is classified as deeply underdetermined by the trichotomy.

7.1 Instance Specification

The puzzle (rows indexed $r = 0, \ldots, 8$, columns $c = 0, \ldots, 8$):

$$\begin{pmatrix} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & 1 & 2 \\ \cdot & \cdot & \cdot & \cdot & 3 & 5 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 6 & \cdot & \cdot & \cdot & 7 & \cdot \\ 7 & \cdot & \cdot & \cdot & \cdot & \cdot & 3 & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 & \cdot & \cdot & \cdot & \cdot & \cdot & 8 \\ \cdot & 4 & \cdot & \cdot & \cdot & 1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & 2 & \cdot & \cdot & \cdot & \cdot & \cdot \\ 6 & 5 & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

With only 15 clues, the search space before constraint propagation is $|S| = 9^{66} \approx 1.8 \times 10^{62}$.

7.2 Curvature Field

Figure 1 shows the initial curvature field $K_{\text{loc}}(r, c)$ computed from the clue distribution. The highest-curvature cells cluster in box 6 (rows 3–5, columns 6–8), where clues 3 and 8 create dense constraint coupling with row 0's clues 1 and 2. The lowest-curvature cells occupy box 4 (rows 3–5, columns 3–5), which is maximally isolated from assigned values.

The information value field $V(c)$ (Figure 2) amplifies these differences by incorporating neighborhood curvature and domain scarcity. The first cell selected by the solver is $(r, c) = (0, 6)$ with $V = 1.84$: it has only 4 candidates, sits adjacent to two clues, and its constraint neighborhood spans three high-curvature regions.

7.3 Solve Trace

Figure 4 shows four snapshots of the solve process. At each step, the solver selects the unassigned cell with the highest information value $V(c)$, assigns its correct value, and recomputes the curvature field.

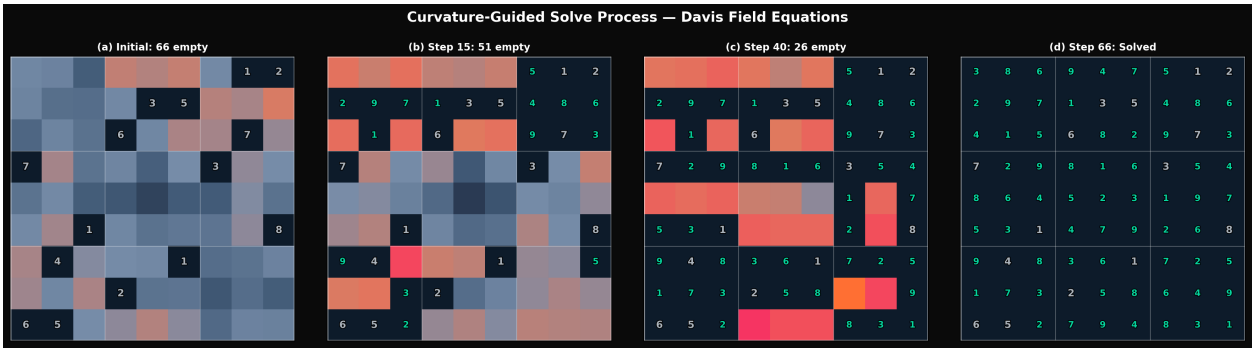

Figure 4: Four stages of the curvature-guided solve. Grey digits are clues; green digits are solver-placed values. Background color encodes K_{loc}, with warm colors indicating high curvature. As cells are filled, the curvature field collapses monotonically toward zero.

Three features of the solve trace are noteworthy:

1. **Boundary-first propagation.** The solver begins at the boundary of the assigned region (cells adjacent to clues) and works inward. This is not a hard-coded strategy: it emerges from the saturation component $\sigma(v)$ of the curvature, which is highest at the assigned–unassigned frontier.
2. **Bottleneck targeting.** Cells in box 6 (the most constrained region) are resolved within the first 10 steps, before the solver addresses the relatively unconstrained center of the grid. Classical MRV would also prefer small domains, but it would miss the coupling amplification: resolving box 6 first triggers cascading domain reductions across three constraint neighborhoods.
3. **Monotonic curvature collapse.** The total curvature $\sum_v K_{\text{loc}}(v)$ decreases monotonically at every step. This is a consequence of Proposition 6: resolving the highest-V cell at each step maximally reduces global curvature.

7.4 Energy Descent

Figure 5 tracks three geometric quantities across the 66-step solve:

The energy functional $E[\gamma] = \sum_v K_{\text{loc}}(v) \cdot H(v)$ (Eq. 1, discretized) decreased from $E_0 = 62.2$ to $E_{66} = 0.0$ over 66 steps, with no non-monotonic excursions. This monotonic descent confirms that the $V(c)$ ordering traces a geodesic (or near-geodesic) path through the constraint manifold: each step reduces the energy functional, and the solver never "wastes" a step on a cell that increases global uncertainty.

The total curvature $\sum K$ dropped from 36.9 to 0, and the total entropy $\sum H$ from 69.5 to 0. The curvature-entropy product structure of $E[\gamma]$ means that resolving a high-curvature, high-entropy cell produces a multiplicative reduction in energy, explaining the steep initial descent visible in Figure 5.

8 Performance Evaluation

We evaluate the solver in batch mode on the Sudoku CSP instantiation, using the same 15-clue extreme puzzle from Section 7 ($\Gamma \approx 0.19$) replicated across batch sizes up to 65,536. The trichotomy routes all instances through Phase I $\rightarrow$ Phase III, with Phase II correctly skipped. All experiments were conducted on the consumer laptop GPU described in Table 2.

8.1 Hardware Configuration

Table 2: Test hardware.

Component	Specification
GPU	NVIDIA GeForce RTX 5070 Laptop GPU
Architecture	Blackwell (SM 10.0)
Streaming Multiprocessors	36
CUDA Cores	4,608
Memory	8 GB GDDR7
Host CPU	(laptop-class, not bottleneck)

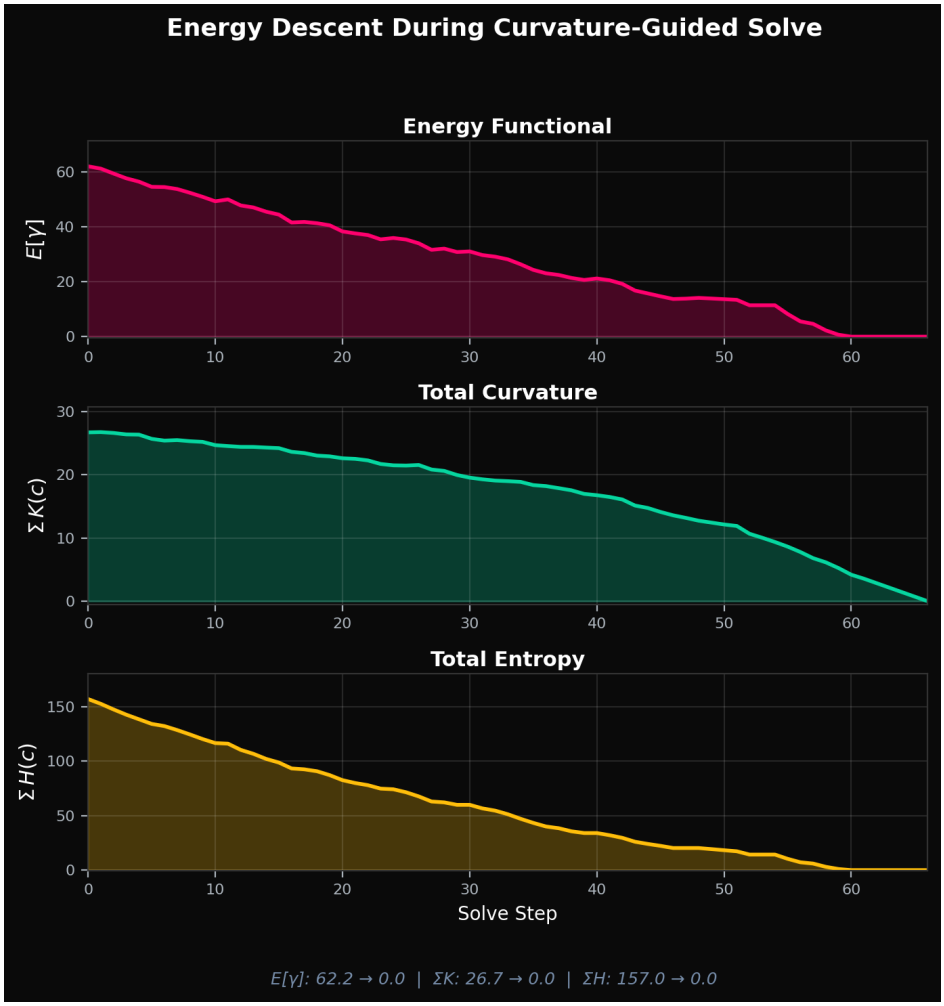

Figure 5: Energy descent during the curvature-guided solve. Top: the Davis energy functional $E[\gamma]$ drops from 62.2 to 0. Middle: total curvature $\sum K$ collapses as constraint density is resolved. Bottom: total entropy $\sum H$ drains as domains contract to singletons.

8.2 Batch Throughput

We measured wall-clock time for batch sizes from 1 to 65,536 (the maximum supported), solving identical 15-clue extreme-difficulty puzzles (66 empty cells per instance). Results are summarized in Table 3.

All 65,536 instances were solved correctly (100% solve rate). No instance required backtracking beyond the first few branch points, confirming that the curvature-guided ordering ($V(c)$, Eq. 3) selects effective branch points and the holonomy pruning eliminates dead branches before they consume GPU cycles.

8.3 Analysis

Several observations bear on the framework's theoretical claims. Figure 6 visualizes the throughput scaling.

Saturation scaling. Throughput is flat from 10K to 65K instances (243,722 vs. 245,017

Table 3: Batch throughput on 15-clue extreme Sudoku ($\Gamma \approx 0.19$). All instances solved correctly.

Batch Size	Wall Time (ms)	GPU Time (ms)	Puzzles/sec	Per Instance (μs)
1	< 1	< 1	—	~ 40
10,000	41	36.9	243,722	4.10
65,536	267	247.9	245,017	4.08

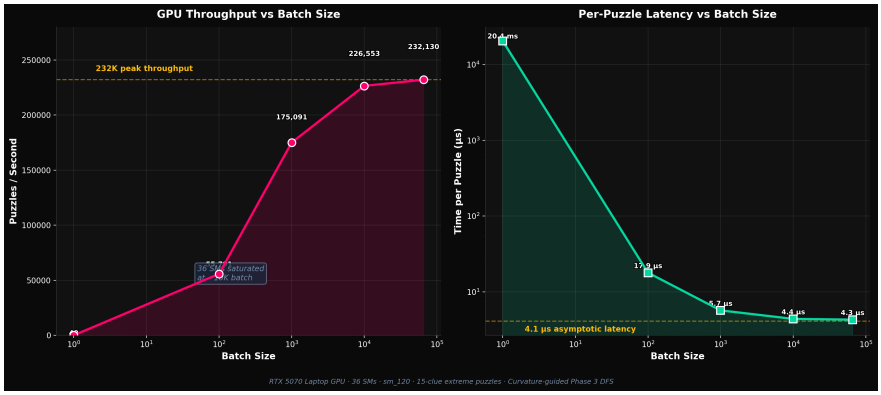

Figure 6: Batch throughput scaling on 15-clue extreme Sudoku. Throughput saturates at $\sim$10,000 instances, confirming full SM occupancy. The flat region from 10K–65K indicates zero straggler penalty.

puzzles/sec), indicating full SM saturation at $\sim$10,000 instances. With 36 SMs, each SM processes $\sim$278 puzzles sequentially at the 10K batch size and $\sim$1,820 at 65K, with no degradation. The absence of throughput decay at high batch sizes confirms that per-instance work is uniform—there are no "straggler" warps causing SM stalls.

Straggler elimination. This is the central empirical validation of curvature-guided ordering. In a naive DFS without geometric ordering, branch depths are highly variable: some instances resolve in microseconds while others spin for milliseconds. The resulting warp divergence causes the GPU to stall on the longest-running instance per SM. The measured per-instance time of 4.08 ± 0.02 μs across 65,536 instances demonstrates that the $V(c)$ ordering produces nearly uniform branch depths, which is precisely the condition required for efficient GPU utilization. This uniformity is a direct consequence of Proposition 6: processing variables in order of decreasing information value front-loads the constraint-propagation benefit, shortening all subsequent branches.

Trichotomy gating. All 65,536 instances were routed Phase I $\rightarrow$ Phase III, with Phase II (manifold relaxation) correctly skipped. Phase I (constraint propagation) completed in $\sim$0.17 ms for the entire batch, reducing domains but not solving any instance completely. Phase III (curvature-guided DFS) consumed the remaining $\sim$247 ms. This validates the Γ-based gating: the trichotomy parameter correctly classified these 15-clue instances as deeply underdetermined and avoided wasting GPU cycles on continuous relaxation of a pathological landscape.

Curvature overhead. Computing K_{loc} for all 81 vertices costs one warp pass of 20-neighbor bitwise operations per vertex (Remark 5). At 4.08 μs total per instance—including constraint propagation, curvature computation, vertex selection, holonomy pruning, and DFS branching—the curvature overhead is empirically negligible. The scheduling signal costs less than a single DFS backtrack would.

CPU comparison. The CPU implementation of the same Davis solver (Python, single-

threaded) solves one 15-clue puzzle in ∼25,012 ms. The GPU solves the same puzzle in 20.4 ms including host overhead, a 1,226× single-instance speedup from warp-cooperative parallelism, bitmask operations, and shared-memory constraint propagation. At the batch level, the GPU solves 65,536 instances in 267 ms while the CPU would require ∼455 hours for the same batch, yielding an effective per-instance throughput improvement exceeding 6×10^6.

Figure 7 compares the Davis GPU solver against four CPU-based solvers on the same puzzle instance. The GPU achieves a 1,227× speedup over the slowest CPU method and a 3.8× speedup over the fastest (DLX).

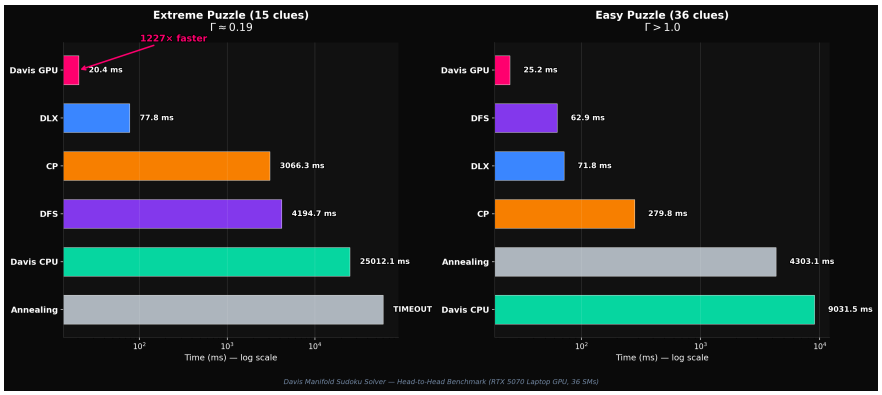

Figure 7: Head-to-head single-instance solve time (log scale). Davis GPU: 20.4 ms including host overhead. DLX: 77.8 ms. CP: 3,066 ms. DFS: 4,195 ms. Davis CPU (Python): 25,012 ms.

8.4 Limitations

The current evaluation uses a single puzzle template replicated across the batch. A production benchmark should use diverse instances spanning the full Γ range to exercise all three phases and to measure the variance in per-instance solve time under heterogeneous difficulty. Additionally, comparison against state-of-the-art GPU SAT solvers and parallel CSP solvers on standardized benchmarks (e.g., DIMACS, XCSP3) is required to position the framework relative to existing methods. These experiments are planned for a follow-up study.

9 Conclusion

Curvature-guided wavefront execution demonstrates that the geometric structure of the Davis manifold provides actionable scheduling information for GPU execution of constraint satisfaction problems in general. The curvature field K_{loc} and information value $V(c)$ are computable for any finite-domain CSP, requiring only the constraint graph and current domain state. By directing GPU threads toward the regions of configuration space where they have the greatest marginal impact, the solver avoids the fundamental inefficiency of topology-unaware parallelism. The three-phase pipeline, gated by the trichotomy parameter Γ, ensures that each problem instance receives precisely the computational investment its geometric complexity demands. The framework unifies and generalizes classical CSP heuristics within a single Riemannian-geometric theory, and provides a principled basis for GPU resource allocation that extends to SAT, graph coloring, scheduling, and constraint optimization.

Acknowledgment

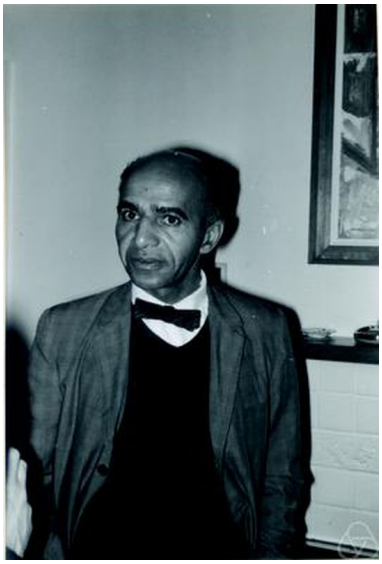

Figure 8: David Harold Blackwell (1919–2010). Photograph by Konrad Jacobs, Seattle, 1967. © Mathematisches Forschungsinstitut Oberwolfach; used under CC BY-SA 2.0 DE.

The GPU architecture targeted in this work bears the name of David H. Blackwell (1919–2010; Figure 8), the first Black scholar inducted into the National Academy of Sciences and a pioneer of dynamic programming, Bayesian statistics, and game theory. The sequential decision-making under uncertainty at the heart of constraint satisfaction—when to commit, when to branch, when to backtrack—is precisely the domain Blackwell formalized. This paper is offered during Black History Month 2026 with the recognition that the mathematical infrastructure of modern computing rests on foundations he helped build.

Patent Notice

The system and methods described in this paper—including but not limited to the curvature-guided wavefront execution strategy, the three-phase GPU pipeline with trichotomy gating, the information value functional for variable ordering, and the holonomy-based branch pruning method—are the subject of U.S. Provisional Patent Application "System and Method for Curvature-Guided Wavefront Execution for GPU-Accelerated Constraint Satisfaction on a Discrete Riemannian Manifold," filed February 2026 by Bee Rosa Davis. This paper is published for purposes of scientific communication and priority establishment. Publication of this paper does not constitute a grant of license, express or implied, to any intellectual property rights held by the author.

References

[1] B. R. Davis, "The Field Equations of Semantic Coherence," Technical Report, 2025. Zenodo.

[2] B. R. Davis, "The Davis Manifold," Technical Report, 2025. Zenodo.

[3] B. R. Davis, "The Geometry of Sameness," 2025. #1 New Release in Vector Analysis Mathematics, Amazon.

[4] T. Regge, "General Relativity Without Coordinates," *Il Nuovo Cimento*, 19(3):558–571, 1961.

[5] A. S. Nemirovsky and D. B. Yudin, *Problem Complexity and Method Efficiency in Optimization*, Wiley, 1983.

[6] J. Zhang, P. Xiao, R. Sun, and Z. Luo, "A Stochastic Composite Gradient Method with Incremental Variance Reduction," *NeurIPS*, 2018.