Faster Exponential-Time Approximate Counting via Bounded Self-Reductions

Katie Clinch
University of Queensland, Brisbane, Australia

Serge Gaspers
UNSW Sydney, Australia

Simon Mackenzie
UNSW Sydney, Australia

Qi Wang
UNSW Sydney, Australia

Abstract

We give faster exponential-time randomised approximation algorithms for counting problems where polynomial-time approximation is unavailable and exact exponential-time counting remains expensive. For general n-vertex graphs, our independent-set counter runs in $O^*(1.1869^n)$ time, improving the previous $O^*(1.2041^n)$ general-graph bound. For n-variable #2-SAT, we obtain an $O^*(1.2373^n)$-time approximation algorithm, narrowly below Wahlström's currently cited $O^*(1.2377^n)$ variable-parameter exact bound.

The new algorithmic point is to take the square root after decomposition. For a single bounded unweighted self-reduction with $f(x)$ positive leaves and recursion-compatible upper bound $b(x)$, an enumerate-or-sample estimator gives an (ε, δ)-approximation in

$$O^*\left(\sqrt{b(x)}\,\varepsilon^{-2}\log\tfrac{1}{\delta}\right)$$

time. After preprocessing decomposes an input into many bounded cores, the combined estimator pays

$$O^*\left(\sqrt{\sum_i b_i(x_i)}\,\varepsilon^{-2}\log\tfrac{1}{\delta}\right),$$

rather than estimating the cores separately at cost $\sum_i \sqrt{b_i(x_i)}$.

The same conversion improves the bases for counting maximal cliques, minimal separators, and perfect matchings in subcubic graphs. Bounded unweighted self-reductions provide the formal language; at the level of counting classes, the resulting unweighted formulation has the same Karp closure as TotP. With explicit recursion-tree access, the framework yields black-box quantum speed-ups.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Approximate counting, exponential-time algorithms, randomised algorithms, #Independent-Set, #2-SAT, self-reducibility, TotP

Digital Object Identifier 10.4230/LIPIcs.ESA.2026.104

Related Version *Full Version*: `https://arxiv.org/abs/2607.06393` [3]

Funding *Serge Gaspers*: acknowledges funding by the Australian Research Council, project DP210103849.

1 Introduction and Results

Many natural counting problems sit in a regime where a polynomial-time multiplicative approximation would be surprising, but exact exponential-time counting is still a very demanding benchmark. This includes #2-SAT, #Independent-Set, and #Maximal-Clique, via approximation-preserving reductions from #SAT [4, 6]. For these problems the

fine-grained question is how much of the exact exponential cost can be avoided while still obtaining a rigorous randomised approximation. Enumeration alone does not answer this question: an enumerator pays for every solution it prints, while an approximate counter only needs enough access to the solution space to estimate its size.

The access we use is already implicit in many exact exponential-time algorithms. A branching algorithm defines a recursion tree whose accepting leaves are the objects being counted. Its analysis often proves more than a running-time bound at the root: it gives an explicit upper bound on the number, or weighted mass, of accepting leaves below every residual instance. The idea of this paper is to turn that bound from a certificate used after the fact into the distribution from which the approximate counter samples. Search-tree analysis then becomes a reusable approximation primitive.

The examples in Table 1 are chosen to test the idea across this landscape. Independent-set counting is a central graph problem and, in statistical-physics language, the hard-core partition function; the approximation tools that we use on easy residuals come from the correlation-decay and spatial-mixing literature [15, 11, 7]. Those tools are strongest in structured regimes, whereas the input here is an arbitrary graph; the role of preprocessing is to expose exactly the residuals where that theory can be used and to aggregate the rest. For richer CNF languages, known exponential-time approximation schemes already beat exact counting in regimes such as #3-SAT [13, 10]; in propositional model counting more broadly, universal-hashing based counters are a central paradigm [2]. #2-SAT is the awkward middle case: it is still resistant to polynomial-time multiplicative approximation, but too restricted to inherit those richer-CNF speed-ups directly. Maximal cliques, minimal separators, and perfect matchings then test whether the same mechanism survives outside these two headline applications.

This paper gives a general method for converting search-tree upper bounds into faster exponential-time approximate counters. For fixed accuracy and confidence, we approximate the number of independent sets in an n-vertex graph in $O^*(1.1869^n)$[1] time, improving the general-graph $O^*(1.2041^n)$ bound of Goldberg, Lapinskas and Richerby [7]. For #2-SAT, we approximate the satisfying assignments of a 2-CNF formula on n variables in $O^*(1.2373^n)$ time, below the currently cited $O^*(1.2377^n)$ variable-parameter exact worst-case bound of Wahlström [14]. Further direct applications give the bounds in Table 1 for maximal cliques, minimal separators, and perfect matchings in subcubic graphs.

For #2-SAT, the numerical gap is narrow but the comparison is meaningful. To our knowledge this is the first n-variable exponential-time approximation bound for general #2-SAT below the best published exact n-variable base. The hybrid uses the part of Wahlström's analysis that exposes residual formulae as a bounded recursion: branches add over residual instances, infeasible residuals are recognised in polynomial time, and the measure gives a usable bound on the hard leaves. Later Wahlström-type improvements with the same exposed structure can be inserted into the same balance calculation; algorithms based on algebraic cancellation, memoisation, or global dynamic programming would first need to be expressed in this bounded-recursion interface.

The main mechanism is a square root taken at the right moment. For one recursion tree with a recursion-compatible upper bound $b(x)$ on the number of positive leaves, an enumerate-or-sample estimator costs $O^*(\sqrt{b(x)})$. The stronger theorem applies when preprocessing

[1] $O^*(\cdot)$ suppresses factors polynomial in the input size; dependence on accuracy and confidence is displayed where it is relevant.

Table 1 Classical running times obtained in this paper. The mechanism column records where the square-root estimator is used. Baselines marked by † are exact-counting or enumeration-based exact baselines.

Problem	This work	Baseline	Mechanism
#Independent-Set	$O^*(1.1869^n)$	$O^*(1.2041^n)$ [7]	sum cores + FPRAS
#2-SAT	$O^*(1.2373^n)$	$O^*(1.2377^n)^\dagger$ [14]	hybrid sum cores
#Maximal-Clique	$O^*(3^{n/6})$	$O^*(3^{n/3})^\dagger$ [8, 1]	single bound
#Minimal-Separator	$O^*(1.2721^n)$	$O^*(1.6181^n)^\dagger$ [12]	single bound
#Perfect-Matching, $\Delta \leq 3$	$O^*(1.1611^n)$	$O^*(1.1918^n)^\dagger$ [5]	unweighted core

leaves many hard cores $x_1, \ldots, x_\ell$, with bounds $b_i(x_i)$. Running the single-core estimator on each core would cost $\sum_i O^*(\sqrt{b_i(x_i)})$. Instead, the sampler ranges over the disjoint union of all bound masses and pays

$$O^*\left(\sqrt{\sum_{i=1}^{\ell} b_i(x_i)}\right),$$

up to polynomial, accuracy, and confidence factors.

The headline applications use this at two levels. Locally, a residual instance decomposes additively into children, with accepting and rejecting leaves; globally, preprocessing branches until a residual is either easy for known reasons or small enough to become a bounded hard core. The local estimator gives a square root of one bound, while preprocessing without aggregation would leave exponentially many residuals to estimate separately.

This interface makes exact-search analyses usable in a new way. A branching recurrence is usually treated as a certificate for an exact running time and then left behind; here the same recurrence, provided it is available at every residual instance, becomes a sampling rule. The method avoids problem-specific near-uniform samplers for independent sets, 2-SAT assignments, and maximal cliques. It uses the structural information that exact exponential-time algorithms already tend to expose. The full version of this paper contains the formal bounded-self-reduction development and all deferred application proofs [3].

The following pieces organise the paper. Recurrence and extremal upper bounds become sampling distributions for one bounded recursion. Preprocessing peels off easy subcases and leaves bounded hard kernels. The sum-of-cores theorem then recombines those kernels before sampling. The exponents come from balancing two explicit quantities: how many residuals preprocessing can create, and how much certified bound mass those residuals carry.

The formal abstraction is deliberately generic. We use bounded unweighted self-reductions: internal nodes contribute the sum of their children, leaves contribute 0 or 1, and a computable bound behaves like a standard recurrence upper bound. This interface exposes the common structure behind self-reducibility, enumeration, exact search-tree bounds, and randomised approximation. That genericity is useful even for the concrete applications: problem-specific preprocessing creates residual graphs or formulae, and the same estimator recombines them once they carry compatible bounds. At the level of counting classes the interface recovers TotP up to Karp closure, but the running-time gains in Table 1 come from keeping the quantitative recurrence information through this decomposition and recomposition.

2 The Core Estimate: Enumerate or Sample

We start with one bounded recursion. The input x unfolds into a polynomial-depth recursion tree for a counting function f. Each internal node y has polynomially many children $h(y,1),\dots,h(y,r(y))$, and each leaf contributes $t(y)\in\{0,1\}$. Thus $f(x)$ is the number of positive leaves below the root. The algorithm only needs local access: it can generate the children of a node, test leaf values, and enumerate positive leaves with polynomial delay. In applications this enumeration is supplied either directly by a solution-enumerating procedure or by the usual self-reduction traversal with a polynomial-time feasibility test for whether a residual subtree contains a positive leaf. Feasibility is only used to handle the small-count case exactly. If the number of positive leaves is below the enumeration threshold, the algorithm finds them all and returns the exact value, including zero. Once the threshold is reached, feasibility no longer plays a role; the algorithm samples from the upper bound itself.

▶ **Definition 2.1** (Bounded recursion tree). *A bounded recursion tree for $f(x)$ is a polynomial-depth, polynomial-fan-out recursion tree with locally computable children $h(y,i)$, leaf values $t(y)\in\{0,1\}$, and $f(x)$ equal to the sum of positive leaves. It comes with a computable non-negative bound b such that positive leaves have $b(y)=1$ and every internal node satisfies*

$$\sum_{i=1}^{r(y)} b(h(y,i)) \le b(y).$$

Positive leaves are enumerable with polynomial delay, using a feasibility or zero test when the application needs one.

This is the only structure used by the estimator. Branching recurrences, extremal solution-counting theorems, and the hard-core bounds used below all produce upper bounds of this form. It is best to regard b as non-negative mass. The children receive at most the mass of their parent, and any unused mass is failure mass. This matches how the bounds arise in practice: branching recurrences and measure-and-conquer analyses usually produce real-valued bounds. The estimator needs these values to be computable, non-negative, and compatible with the recursion inequality; it does not need them to count an artificial set of leaves. The full version implements the finite distribution with integer ticket bounds; this is just a discretisation of the same mass view, and the displayed real bases are safe asymptotic summaries of such compatible bounds.

▶ **Theorem 2.2** (Single-core estimator; new). *Under Definition 2.1, there is a randomised algorithm which, on input x, returns an (ε,δ)-approximation to $f(x)$ in time*

$$O^*\left(\sqrt{b(x)}\,\varepsilon^{-2}\log\tfrac{1}{\delta}\right).$$

Proof. Let $B=b(x)$ and $k=\lceil\sqrt{B}\rceil$. The estimator first enumerates positive leaves, stopping after k outputs. If enumeration finishes first, every positive leaf has been seen and the algorithm returns the exact value. This includes $f(x)=0$, which later lets us discard zero-valued residuals without damaging a relative-error guarantee. If instead k leaves are found, then $f(x)\ge k$, and the count is large enough to estimate by random samples from the bound mass. The enumeration step is what turns the sampler into a multiplicative estimator. Without it, a tiny value of $f(x)$ could make the success probability $f(x)/B$ too small for square-root running time.

The sampling distribution is obtained from the recursion and the bound, not from an exact count of feasible leaves. At an internal node y, move to child $h(y,i)$ with probability $b(h(y,i))/b(y)$; with the remaining probability, if any, stop at a dummy failure. The inequality

defining the bound ensures that these probabilities sum to at most one. Along any root-to-leaf path the probabilities telescope, so each unit of positive leaf mass is reached with probability $1/B$. Dummy failures and zero leaves account for the slack between the bound and the true count. Thus one sample succeeds with probability exactly $f(x)/B$. The sampler stays local throughout: it asks only for a child's recurrence bound, not for the number of successful leaves below that child. Hard residuals can therefore be sampled by walking through the same branching structure as an exact algorithm, with recursive exact counts replaced by probabilistic choices proportional to certified upper bounds.

After the enumeration stage, in the sampling case, $f(x)/B \geq 1/\sqrt{B}$. Standard Chernoff bounds therefore give a relative-error estimate using $O(\sqrt{B}\,\varepsilon^{-2}\log(1/\delta))$ independent samples. All polynomial factors for traversing the recursion, drawing from the child distribution, and amplifying confidence are absorbed by $O^*(\cdot)$. The theorem acts as a compiler for upper bounds: a recursion with at most B positive leaves in this compatible sense has a randomised approximate counter with square-root dependence on B. Ordinary polynomial overheads for child generation, leaf testing, feasibility testing, or evaluating the bound remain hidden in O^*, so standard graph and formula preprocessing does not change the displayed bases. ◀

For some problems, an extremal theorem already supplies the bound. For the two main rows of Table 1, the needed bound appears only after the input has been decomposed into many residual instances, and the number of residuals is itself exponential.

3 The Aggregate Square-Root Principle

Preprocessing may leave exponentially many hard residual instances. If residual core i has bound $B_i = b_i(x_i)$, then applying Theorem 2.2 independently to all cores would cost

$$\sum_i O^*(\sqrt{B_i}),$$

with separate accuracy and confidence bookkeeping for every estimate.

The aggregate theorem builds one estimator for the disjoint union of all hard cores. Easy contributions may be handled separately and added afterwards; zero-valued hard cores are discarded using the application-specific feasibility or zero-test before the aggregate estimator is invoked. The square root is then taken after summing the residual bounds. Because all contributions are non-negative, this directly estimates the total hard contribution. No core has to be accurately estimated in isolation, and no separate failure probability has to be assigned to each residual. This matters when some cores have tiny counts: such cores can be expensive to estimate alone but harmless in aggregate.

Operationally, the family of hard cores becomes one larger forest. The new root chooses a core according to its bound mass, then the local bound sampler runs inside that core. After this first choice, the construction is exactly the single-core estimator with total bound $B = \sum_i B_i$.

▶ **Theorem 3.1** (Sum of cores; new). *Suppose a preprocessing algorithm, in time S, constructs hard cores $x_1, \ldots, x_\ell$, each equipped with Definition 2.1 and bound $B_i = b_i(x_i)$, after discarding zero-valued cores. Then*

$$F := \sum_{i=1}^{\ell} f_i(x_i)$$

has an (ε, δ)-approximation computable in time

$$S + O^*\left(\sqrt{\sum_{i=1}^{\ell} B_i\,\varepsilon^{-2}\log\tfrac{1}{\delta}}\right).$$

Proof. Let $B = \sum_i B_i$. The estimator first enumerates positive leaves over the disjoint union of all core trees until either every core is exhausted or $\lceil \sqrt{B} \rceil$ positive leaves have been found. In the first case the total hard contribution F is known exactly. In the second case $F \geq \sqrt{B}$, so the combined success probability is large enough for sampling. Enumeration over the union can be implemented by dovetailing the positive-leaf enumerators for the nonzero cores. A core is fully exhausted only when the total number of positive leaves in the whole union is small; the algorithm may store the residual cores and their bounds, but it cannot afford a separate square-root estimate for each one.

For one sample, choose core i with probability B_i/B, and then run the single-core bound sampler inside that core. The success probability is

$$\sum_i \frac{B_i}{B} \cdot \frac{f_i(x_i)}{B_i} = \frac{\sum_i f_i(x_i)}{\sum_i B_i} = \frac{F}{B}.$$

Thus the same Chernoff calculation as in Theorem 2.2 gives the claimed dependence on $\sqrt{B} = \sqrt{\sum_i B_i}$. Uniform sampling over cores would be wrong: small and large bound masses must be weighted according to B_i. The preprocessing phase constructs the list of cores and their bounds, so sampling a core with probability B_i/B can be implemented by the usual prefix-sum or binary-search method, within the polynomial factors hidden by $O^*(\cdot)$. The non-negativity of the summands is used in two places. First, the weighted mixture has success probability F/B without any cancellation. Second, once the hard contribution has been estimated, it can be added to the easy contribution estimated by other means while preserving a multiplicative guarantee after the usual accuracy allocation. The setting naturally fits counting problems whose self-reductions express the answer as a sum of residual answers. ◀

The distinction is exponential in the regimes we use. If preprocessing creates at most $2^{\alpha n}$ hard cores and each has bound at most $2^{\alpha n}$, then the aggregate mass is at most $2^{2\alpha n}$ and the combined estimator costs $O^*(2^{\alpha n})$. Estimating the same cores separately could cost $2^{\alpha n} \cdot 2^{\alpha n/2} = 2^{3\alpha n/2}$. The applications below are engineered so that the preprocessing frontier and the hard-core estimator balance only after the residual bounds have been summed. Preprocessing creates a family of residual instances whose individual counts may still be hard, but whose total certified mass is small enough for aggregate sampling.

4 Applications

The applications fall into two groups. For extremal-output problems, an existing solution bound already serves as a recursion-compatible leaf bound, so Theorem 2.2 applies almost directly. For #Independent-Set and #2-SAT, the global recursion is too weak. The algorithm first removes easy residuals and exposes bounded hard cores; Theorem 3.1 then estimates the total hard contribution.

The problem-specific work is concentrated in four checkable questions: which residuals are easy, which residuals remain hard, what bound controls each hard residual, and how large the preprocessing frontier can be. Once these quantities are proved, the same estimator supplies the recomposition step. The square-root sampler is not retuned for graph instances or formula instances; it only sees residual instances with certified bounds.

4.1 Direct Extremal Applications

Counting maximal cliques is the simplest direct example. The Bron–Kerbosch solution-enumerating recursion supplies the search tree, and the Moon–Moser theorem bounds the number of maximal cliques in an n-vertex graph by $3^{n/3}$ [8, 1]. Using this as b, Theorem 2.2 gives an (ε, δ)-approximation in $O^*(3^{n/6}) = O^*(1.2009^n)$ time. A worked verification of the bounded recursion is given in Appendix A. The same direct use of a solution bound applies to minimal separators, via the golden-ratio extremal bound, and to perfect matchings in subcubic graphs once the exact branching rule is written as an unweighted recursion. The corresponding bounded-recursion presentations and arithmetic are given in the full version [3]. For minimal separators, the exact enumeration bound is $O^*(\varphi^n)$, and the estimator therefore gives $O^*(\sqrt{\varphi}^n) = O^*(1.2721^n)$. For perfect matchings in subcubic graphs, the bounded recursion in the full version has certified leaf bound $O^*(1.3481^n)$, yielding $O^*(1.1611^n)$ after the same square-root conversion. The arithmetic in these direct rows is therefore easy to audit: the estimator receives $3^{n/3}$, φ^n, and 1.3481^n as certified leaf bounds and returns their square-root bases. For maximal cliques, the extremal theorem is tight and the square-root conversion is transparent. For minimal separators, the full version also has to organise the enumeration so that the residual completion test remains polynomial-time. For subcubic perfect matchings, the issue is different again: exact algorithms are often phrased with weighted or multiplicative branching, while the interface here asks for an unweighted leaf-counting recursion. Once that interface is exposed, however, the estimator itself is unchanged.

4.2 Counting Independent Sets

Independent Set is the main decomposition application. A global include/exclude recursion has the correct unweighted form, but the trivial bound 2^n would only give an $O^*(2^{n/2})$-scale estimator. The algorithm therefore uses this recursion only after preprocessing has isolated small hard residuals. This is a useful stress test for the framework. The easy side relies on specialised FPRAS machinery for the hard-core model, while the hard side is recombined by the problem-independent aggregate theorem. Thus the improvement is not a new correlation-decay argument; it is a generic way of using that theory inside a faster general-graph exponential-time approximation algorithm.

Preprocessing branches on high-degree structure and stops at two kinds of frontier node. If the residual graph enters the known low-degree/low-connective-constant regime used by the independent-set FPRAS machinery, it is handled as an easy instance [15, 11, 7]. Concretely, the full analysis uses the condition that every vertex of degree at least 6 has 2-degree at most 26, where the 2-degree of a vertex is the sum of the degrees of its neighbours. Otherwise, when the deleted-vertex budget is exhausted, the residual graph is kept as a hard core. Preprocessing removes residuals where the known FPRAS machinery applies and passes only the remaining kernels to the aggregate estimator. The improvement over the square root of the trivial 2^n recursion comes from meeting two controls at the same scale: a structural easy regime and a deleted-vertex cutoff for the hard side. The branching itself is the standard independent-set split. For a chosen vertex v, every independent set either excludes v, giving an independent set of $G - v$, or includes v, in which case the whole closed neighbourhood $N[v]$ is deleted. High-degree vertices are attractive because the include branch deletes many vertices, while the exclude branch spends only one unit of the deleted-vertex budget. The full analysis keeps the two controls separate. The local drops used in the proof are summarised by the following vectors, where α is the branching number satisfying $\sum_j \alpha^{-a_j} = 1$:

trigger	drop vector	$\log_2 \alpha$
$\deg(v) \geq 7$, measure	$(1, 8)$	0.301066
$\deg(v) = 6,\ \deg^2(v) \geq 27$, measure	$(2, 5.5)$	0.289994
deleted-budget tree bound	$(1, 7)$	0.328174

The displayed frontier bound uses the conservative deleted-budget row. The measure recurrence bounds the size of the preprocessing tree; the stopping condition bounds the number of live vertices in every hard residual. If the easy condition is reached first, the branch leaves the hard-core calculation. If the deletion cutoff is reached first, the remaining graph has the promised size and can be charged by the naive $2^{|V(H)|}$ bound.

With the refined stopping rule and $\beta = 0.7529$, the preprocessing frontier has size at most $2^{0.247083n}$. Every hard residual has at most $(1 - \beta)n = 0.2471n$ vertices. Each such hard residual H is then equipped with the elementary independent-set recursion and the naive bound $b_{\mathrm{IS}}(H) \leq 2^{|V(H)|} \leq 2^{0.2471n}$. Therefore the total hard-core bound mass is at most

$$\sum_H b_{\mathrm{IS}}(H) \leq 2^{0.247083n} \cdot 2^{0.2471n} = 2^{0.494183n}.$$

The hard residuals are then estimated together, not one by one. Theorem 3.1 gives aggregate cost

$$O^*\big(2^{0.247092n}\big) = O^*(1.1869^n),$$

with the aggregate hard side slightly dominating the preprocessing frontier. The easy frontier is summed separately using the cited FPRAS regime, with a standard allocation of the failure probability. Combining the preprocessing, easy leaves, and aggregated hard leaves gives an (ε, δ)-approximation for #Independent-Set in $O^*(1.1869^n)$ time. The running time of the easy side is at most the preprocessing-frontier size times polynomial and accuracy factors, so it is covered by the same $2^{0.247083n}$ scale. The hard side is governed by the square root of the aggregate mass, not by the number of hard cores times the square root of one residual bound. The easy estimates are summed using non-negativity, while the hard contribution is estimated directly as one aggregate quantity by Theorem 3.1; allocating constant fractions of the failure probability to the two sides gives the stated (ε, δ) guarantee. This improves the general-graph $O^*(1.2041^n)$ bound of Goldberg, Lapinskas and Richerby. The input is an arbitrary graph, and the comparison is with the previous general-graph approximation base. The hard residuals are bounded by the naive $2^{|V(H)|}$ count rather than by a sharper specialised theorem. Using this crude bound keeps the core estimator completely generic once the residual size has been proved. Sharper residual bounds would immediately improve the aggregate term, but the naive bound already gives the stated improvement. The camera-ready appendix gives this application as a worked decomposition example.

4.3 Counting 2-SAT Assignments

The #2-SAT row is deliberately included even though the numerical gap is small. The comparison throughout this paragraph is by the number of variables. The open point is whether approximation can buy any worst-case exponential saving for general #2-SAT, not the size of the numerical gap by itself. For #3-SAT and larger clause languages, known approximation schemes can exploit richer structure in the formula [13, 10]; in propositional model counting more broadly, universal-hashing based counters are a powerful general

technology for constraints [2]. At the other end, monotone graph-counting problems have their own FPRAS regimes. General #2-SAT has sat between these sources of speed-up. The hybrid below shows a third route: use exactly the residual recursion that drives the best exact analyses, but aggregate the hard residuals before paying the square root.

A residual 2-SAT state records both the simplified 2-CNF formula and the set of still-unassigned variables. This convention matters because a variable that no longer appears in any clause is still a free choice unless it has been assigned, and must still contribute to the count. Unsatisfiable residuals are discarded using polynomial-time 2-CNF satisfiability. If the residual has maximum degree at most 2, the simple constraint graph is a disjoint union of paths and cycles. After all clauses between each adjacent pair have been folded into the induced binary relation on the two endpoint variables, the corresponding easy residuals are counted by a standard polynomial-time dynamic programme over those paths and cycles, with isolated still-unassigned variables contributing independent factors of two. The hard residuals are those that survive until the fixed-variable cutoff.

The preprocessing is guided by the simple variable-interaction graph on the still-unassigned variables. Two variables are adjacent if some residual clause contains literals on both; degree is therefore not clause multiplicity. For a variable v, let $d_F^+(v)$ and $d_F^-(v)$ count neighbours witnessed by clauses in which v appears positively and negatively, respectively, under a fixed choice of witness clause per neighbour. Branching on v fixes v, and unit propagation fixes at least the neighbours whose witness clauses are falsified. Thus the two branches decrease the number of still-free variables by at least

$$(1 + d_F^+(v),\ 1 + d_F^-(v)).$$

Extra clauses or further propagations can only help. For the upper bound we keep only the neighbours certified by the witness clauses, since that information remains stable under the residual simplifications used in the preprocessing analysis.

For degree 6, the worst polarity split is the one-sided split, giving the safe vector $(7, 1)$; mixed polarities only improve the recurrence. Indeed, a mixed split such as $(4, 4)$ is much stronger for the branching number than $(7, 1)$. The analysis uses the one-sided vector because it is safe without needing a case split over polarity patterns. Once the degree-6 supply can no longer be guaranteed, the specialised preprocessing uses the safe degree-at-least-3 vector $(4, 1)$. It stops after $0.6999n$ variables have been fixed, so every hard residual has at most $0.3001n$ still-free variables. Those hard residuals are aggregated exactly as in Theorem 3.1, using the naive residual bound $2^{0.3001n}$ for each core. As in the independent-set algorithm, the cutoff has two jobs: it limits the depth of the specialised preprocessing tree, and it gives the hard-core size bound needed for the aggregate mass calculation. The naive bound depends on the number of still-free variables recorded in the residual state, not on the number of remaining clauses. The persistence statement behind the degree-6 phase is local. If the root residual has many degree-6 variables and maximum degree at most 6, then fixing one variable can spoil only that variable and its at most six root neighbours as possible future degree-6 branch vertices. Thus enough degree-6 variables remain available for the initial part of the preprocessing tree.

The global hybrid first branches away variables of degree at least 7, using the safe vector $(8, 1)$. Once all residuals have maximum degree at most 6, we split according to the number of variables of degree below 6. If there are at most an $\eta^\star$-fraction, we use the specialised degree-6-heavy aggregate routine. Otherwise we invoke Wahlström's exact degree-measure algorithm on the residual. The specialised preprocessing has exponent

$$0.305885 + 0.019541\,\eta^\star.$$

In the complementary case, using Wahlström's degree weights $w_6 = 0.307612$ and $w_5 = 0.301245$, the exact term is at most

$$(1-\eta^\star)w_6 + \eta^\star w_5.$$

The switch point is the solution of

$$0.305885 + 0.019541\,\eta = (1-\eta)w_6 + \eta w_5,$$

which gives $\eta^\star \approx 0.0667$ and worst-case exponent below 0.30719. Hence the hybrid approximate counter runs in

$$O^*(2^{0.30719n}) = O^*(1.2373^n),$$

slightly below the currently cited $O^*(2^{0.307612n}) = O^*(1.2377^n)$ variable-parameter exact worst-case bound. The two regimes cover complementary structure. When the residual is degree-6-heavy, the specialised preprocessing tree is small enough, and the aggregate estimator for its hard cores is cheaper than the preprocessing. When too many variables have degree below 6, Wahlström's measure gains from the lower w_5 weight and improves on the all-degree-6 worst case. The displayed value of $\eta^\star$ is where these two explanations meet.

The modularity is partial. Our theorems accelerate the part of an exact analysis that can be viewed as bounded branching over residual instances. If a future Wahlström-type improvement keeps the residual structure described in the introduction visible, then its measure term can be substituted into the hybrid and rebalanced. An algorithm whose advantage comes from algebraic cancellation, memoisation, state merging, or global dynamic programming would need a separate reduction to this interface before the theorem applies. Recent clause-parameter algorithms for weighted #2-SAT and #3-SAT point to a complementary direction [9]: if such analyses can be refactored into compatible bounded recursions measured by clauses, they become natural candidates for the same aggregation principle, but they are not the variable-parameter comparison made above.

5 Scope and Further Consequences

The abstraction used above is the bounded recursion tree of Definition 2.1, formalised in the full version through bounded unweighted self-reductions. It isolates the pieces shared by self-reducibility, enumeration, exact exponential-time bounds, and randomised approximation. At the level of counting classes, the corresponding unweighted formulation has the same Karp closure as TotP. The complexity-class statement is conceptual scope; the running-time gains come from the explicit bounds and decompositions in the applications. The value of the generic formulation is that it permits local problem-specific work and global recomposition to be separated. Enumeration and approximate counting are no longer separate primitives: enumeration explores positive leaves of the recursion tree, while the estimator samples from a bound on that same tree. In this sense, TotP serves less as a membership label and more as a guide to the right algorithmic abstraction. Self-reducibility supplies the residual subproblems, enumeration supplies the small-count exact case, and the explicit recurrence bound supplies the sampling measure. Because every hard residual remains an object of the same bounded-recursion type, problem-specific preprocessing can be combined with the problem-independent estimator; the sum-of-cores theorem is exactly this recomposition step.

This local tree interface supports nontrivial black-box quantum variants. Because self-reducibility gives access to child generation, feasibility information, bounded depth, and the recurrence bound, quantum routines can act on the traversal or tree-size side as well as on the

final sampling side. In particular, standard quantum approximate counting gives a generic $b(x)^{1/3}$ dependence, and adding rooted-tree access for Ambainis–Kokainis tree-size estimation gives $b(x)^{1/4}$ for bounded uSR instances. For the decomposition-heavy Independent Set and #2-SAT rows, the leading cost in this paper remains the classical preprocessing tree, so the full version treats quantum speed-ups as generic consequences of the interface rather than as new headline bases for those applications.

The broader message is not that every exact counting algorithm can simply be "square-rooted". The reusable ingredient is the bounded-recursion structure introduced above: the residual tree is locally accessible, positive leaves can be exposed or ruled out with polynomial overhead, and the tree carries a certified upper bound on its positive leaves. That bound then becomes a sampling distribution. When preprocessing produces many hard residuals, the aggregate theorem sums their bound masses before taking the square root. For maximal cliques, minimal separators, and subcubic perfect matchings, the input is essentially an existing output bound. For Independent Set and #2-SAT, the main work is to create residuals to which such a bound can be applied in aggregate. The same estimator covers both uses; this is the common source of the running-time improvements.

References

1 Coenraad Bron and Joep Kerbosch. Finding all cliques of an undirected graph (algorithm 457). *Commun. ACM*, 16(9):575–576, 1973.

2 Supratik Chakraborty, Kuldeep S. Meel, and Moshe Y. Vardi. Approximate model counting. In Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors, *Handbook of Satisfiability – Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, pages 1015–1045. IOS Press, 2021. doi:10.3233/FAIA201010.

3 Katie Clinch, Serge Gaspers, Simon Mackenzie, and Qi Wang. Faster exponential-time approximate counting via bounded self-reductions, 2026. doi:10.48550/arXiv.2607.06393.

4 Martin Dyer, Leslie Ann Goldberg, Catherine Greenhill, and Mark Jerrum. The relative complexity of approximate counting problems. *Algorithmica*, 38:471–500, 2004. doi:10.1007/S00453-003-1073-Y.

5 Martin Fürer. Counting perfect matchings in graphs of degree 3. In *International Conference on Fun with Algorithms*, pages 189–197. Springer, 2012. doi:10.1007/978-3-642-30347-0_20.

6 Leslie Ann Goldberg, Rob Gysel, and John Lapinskas. Approximately counting locally-optimal structures. *Journal of Computer and System Sciences*, 82(6):1144–1160, 2016. doi:10.1016/J.JCSS.2016.04.001.

7 Leslie Ann Goldberg, John Lapinskas, and David Richerby. Faster exponential-time algorithms for approximately counting independent sets. *Theoretical Computer Science*, 892:48–84, 2021. doi:10.1016/J.TCS.2021.09.009.

8 John W Moon and Leo Moser. On cliques in graphs. *Israel journal of Mathematics*, 3:23–28, 1965.

9 Junqiang Peng, Zimo Sheng, and Mingyu Xiao. New algorithms for #2-SAT and #3-SAT. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, pages 2666–2674, 2025. doi:10.24963/ijcai.2025/297.

10 Manuel Schmitt and Rolf Wanka. Exploiting independent subformulas: A faster approximation scheme for #k-SAT. *Information Processing Letters*, 113(9):337–344, 2013. doi:10.1016/J.IPL.2013.02.013.

11 Alistair Sinclair, Piyush Srivastava, Daniel Štefankovič, and Yitong Yin. Spatial mixing and the connective constant: Optimal bounds. *Probability Theory and Related Fields*, 168(1):153–197, 2017.

12 Ken Takata. Space-optimal, backtracking algorithms to list the minimal vertex separators of a graph. *Discrete Applied Mathematics*, 158(15):1660–1667, 2010. doi:10.1016/J.DAM.2010.05.013.

13 Marc Thurley. An approximation algorithm for #k-SAT. *arXiv preprint arXiv:1107.2001*, 2011. arXiv:1107.2001.

14 Magnus Wahlström. A tighter bound for counting max-weight solutions to 2SAT instances. In *Parameterized and Exact Computation: Third International Workshop, IWPEC 2008, Victoria, Canada, May 14-16, 2008. Proceedings 3*, pages 202–213. Springer, 2008. doi:10.1007/978-3-540-79723-4_19.

15 Dror Weitz. Counting independent sets up to the tree threshold. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 140–149, 2006. doi:10.1145/1132516.1132538.

A A Worked Example: Maximal Cliques

This appendix spells out the direct maximal-clique application. It is the cleanest example of how an existing enumeration tree and an extremal solution bound become a bounded recursion tree in the sense of Definition 2.1.

Let $M(G)$ be the number of maximal cliques of a graph G. Moon and Moser proved the tight extremal bound $M(G) \le 3^{|V(G)|/3}$ [8]. We use the exact Moon–Moser function

$$\mathrm{MM}(t) := \begin{cases} 1, & t \in \{0,1\}, \\ 3^{t/3}, & t \equiv 0 \pmod 3, \\ 4 \cdot 3^{(t-4)/3}, & t \equiv 1 \pmod 3 \text{ and } t \ge 4, \\ 2 \cdot 3^{(t-2)/3}, & t \equiv 2 \pmod 3. \end{cases}$$

This function is nondecreasing, satisfies $t \le \mathrm{MM}(t)$, and is supermultiplicative in the form

$$\mathrm{MM}(a)\mathrm{MM}(b) \le \mathrm{MM}(a+b).$$

Equivalently, $\mathrm{MM}(t)$ is obtained by partitioning t into parts of size 2 and 3, with the usual replacement of $3+1$ by $2+2$.

We use the Bron–Kerbosch recursion with pivoting [1]. A state is a tuple $x = \langle R, P, X, G\rangle$, where R is the current clique, P contains vertices still available to extend R, X contains vertices whose corresponding extensions have already been assigned to earlier branches, and every vertex of $P \cup X$ is adjacent to every vertex of R. The root state is $\langle \emptyset, V(G), \emptyset, G\rangle$. If $P = X = \emptyset$, the state contributes one maximal clique. Otherwise it contributes zero unless it has children.

For a non-leaf state, let $H = G[P \cup X]$, choose a pivot $u \in P \cup X$ of maximum degree in H, and list

$$C = P \setminus N_H(u) = \{v_1, \dots, v_k\}.$$

The children are

$$\langle R \cup \{v_i\}, (P \setminus \{v_1, \dots, v_{i-1}\}) \cap N_G(v_i), (X \cup \{v_1, \dots, v_{i-1}\}) \cap N_G(v_i), G\rangle \qquad (1 \le i \le k).$$

This is the standard Bron–Kerbosch partition: the branches are disjoint and cover all maximal cliques extending R that have not already been charged to X. Therefore the number of maximal cliques of G is the value of this recursion at the root.

Define the state bound

$$b_{\mathrm{MC}}(\langle R, P, X, G\rangle) := \begin{cases} 1, & \text{if } C = \emptyset, \\ \mathrm{MM}(|P| + |X|), & \text{otherwise.} \end{cases}$$

We verify the recursion inequality. Let $x = \langle R, P, X, G\rangle$, let $\Delta = \deg_H(u)$, and let $k = |P \setminus N_H(u)|$. Since u has maximum degree in H,

$$k \le |P| + |X| - \Delta.$$

For the child corresponding to v_i, all remaining possible extension vertices lie in $N_H(v_i)$, so

$$|P_i| + |X_i| \le \deg_H(v_i) \le \Delta.$$

Thus

$$\sum_i b_{\mathrm{MC}}(x_i) \le k\,\mathrm{MM}(\Delta) \le \mathrm{MM}(k)\mathrm{MM}(\Delta) \le \mathrm{MM}(k+\Delta) \le \mathrm{MM}(|P| + |X|) = b_{\mathrm{MC}}(x).$$

The bound is also 1 on positive leaves. Hence this is a bounded recursion tree with root bound at most $3^{n/3}$. Applying Theorem 2.2 gives an (ε, δ)-approximation to #Maximal-Clique in

$$O^*\left(3^{n/6}\varepsilon^{-2}\log\tfrac{1}{\delta}\right) = O^*\left(1.2009^n\varepsilon^{-2}\log\tfrac{1}{\delta}\right)$$

time.

B A Worked Decomposition Example: Independent Sets

The independent-set application illustrates the second use of the framework: preprocessing produces many residual instances, and the square root is taken only after their bound masses have been summed. We spell out the calculation, while leaving the detailed degree-case analysis to the full version [3].

For a graph G, the basic self-reduction is the usual include/exclude split. For a vertex v,

$$Z(G) = Z(G - v) + Z(G - N[v]),$$

where $Z(G)$ denotes the number of independent sets of G. This recursion is a bounded recursion tree with the trivial state bound $2^{|V(G)|}$, but using it directly would give only the $O^*(2^{n/2})$ scale. The point of the application is to expose smaller hard residuals before the generic estimator is used.

The preprocessing branches on high-degree structure. It stops at two types of frontier node. If the residual graph enters the regime handled by known hard-core-model FPRAS machinery, the residual is treated as easy and is counted there. Otherwise, once a fixed deleted-vertex budget has been spent, the remaining graph H is kept as a hard core. At this point no special independent-set structure is used inside H: we attach the elementary include/exclude recursion and the naive bound

$$b_{\mathrm{IS}}(H) \le 2^{|V(H)|}.$$

The two quantities that matter are the number of hard cores and the size of each hard core. With the stopping rule used in the main text, the preprocessing frontier has size at most

$$2^{0.247083n},$$

and every hard core has at most

$$0.2471n$$

vertices. Therefore the aggregate bound mass of all hard cores satisfies

$$\sum_H b_{\mathrm{IS}}(H) \le 2^{0.247083n} \cdot 2^{0.2471n} = 2^{0.494183n}.$$

Applying Theorem 3.1 to this whole family, rather than applying Theorem 2.2 to each hard core separately, gives hard-side cost

$$O^*\left(\sqrt{2^{0.494183n}}\right) = O^*(2^{0.247092n}) = O^*(1.1869^n),$$

up to the usual accuracy and confidence factors.

The distinction from estimating cores independently is the essential point. A separate estimate for each hard core would pay for the frontier size and then again for the square root of an individual residual bound. The aggregate estimator instead samples from the disjoint union of all hard-core bound mass. It chooses a core with probability proportional to $b_{\mathrm{IS}}(H)$, then runs the local bound sampler inside that core. The success probability of one combined sample is the total number of hard independent sets divided by the total bound mass, exactly as in Theorem 3.1. The easy frontier contribution is added separately using non-negativity and a standard allocation of the error and failure budgets.

Thus the independent-set improvement comes from matching two frontiers: the preprocessing tree is small enough, and the remaining hard cores are small enough, that the square root of their total certified mass is below the previous general-graph approximation base. The detailed branching proof in the full version is precisely what certifies the two displayed exponents.