

A guided reading of the matrix construction

The Exact Correction-and-Copy Algorithm

Understanding the construction before worrying about space

Prepared with Gil Cohen

11 October 2026

The central question in these notes is: how can we transfer probability from a continuation matrix into an accumulated reward, while preserving nonnegativity and making every remaining transition small?

We first answer this question with full matrices and no detector. We then add a direct column detector and a copying operation, obtaining a complete, elementary version of the paper's correction-and-copy hierarchy.

We work in exact arithmetic and impose no space bound. Every matrix may be stored explicitly. The examples and proofs are intended to explain the construction itself: its identities, its progress measure, and the reason for each additional ingredient.

The main construction uses a *hard threshold* and a *deterministic column count*. These are deliberate simplifications of the smooth cutoff and sampled detector in Section 4 of the supplied manuscript, *Exact Derandomization of Logarithmic Space: $L = RL = BPL$* , OpenAI, 23 September 2026. Section 10 gives the precise correspondence. The notes prove their simplified hierarchy; they do not supply the small-space implementation of the manuscript.

All diagrams are part of the accompanying, self-contained L^AT_EX source.

Contents

1	Probability as reward plus continuation	1
2	The correction identity and Richardson's equation	3
3	A threshold rule that preserves positivity	4
4	Why a constant number of iterations suffices	7
5	A complete dense algorithm with no detector	10
	Bonus: excess-mass transport, sandpiles, and PageRank	12
6	What goes wrong with unrestricted incoming degree?	15
7	A direct-count detector and its pilot	19
8	Copying the exceptional destinations	24
9	The complete hierarchy and its guarantee	28
10	Returning to the paper: what we simplified and why	32

How to read these notes. Sections 1–5 form a first complete story: the exact dense algorithm. Sections 6–9 explain why the paper adds a detector and copies, and prove a simplified version of that larger construction. An optional bonus after Section 5 relates the dense iteration to divisible sandpiles and PageRank pushes. Section 10 reconnects the notation to the original paper. The bonus is unnumbered, so the ten main section numbers remain unchanged.

Each section starts with its objective and ends by naming what we have accomplished and the objects introduced there. Important terms are emphasized at their first substantive introduction, and occasionally again in a roadmap that brings several players together. The parameter f always denotes an entry scale, and the gate g always acts on columns.

1 Probability as reward plus continuation

Objective

Set up the equation $p = W + Cp$ and give each of its terms a probabilistic meaning. Identify the quantity that the algorithm will eventually make small.

An ordered transition system

Let V be a finite set of states with a fixed ordering. A *transition matrix* C has nonnegative entries and satisfies

$$C(x, y) = 0 \quad \text{unless } x < y, \quad C\mathbf{1} \leq \mathbf{1}. \quad (1)$$

The forward-order condition makes C *strictly forward*. A nonnegative matrix with row sums at most one is called *substochastic*. Thus $C(x, y)$ is the probability of continuing from x to y . A row sum below one allows the process to stop. All inequalities between matrices or vectors in these notes are coordinatewise.

For a finite exact-arithmetic algorithm, take the input entries and rewards to be rational. The algebraic identities also hold for real entries.

The condition $x < y$ says that the graph is *acyclic*. A *layered branching program* is an example: order vertices by their time layer. A *shortcut* introduced later may skip layers, but it will still go forward. Consequently $C^{|V|} = 0$, and

$$(I - C)^{-1} = I + C + \dots + C^{|V|-1}.$$

There is no issue of convergence in this inverse.

Let $W \geq 0$ be a *reward vector*, and let $p(x)$ be the expected total reward collected starting at x . The elementary first-step identity is

$$p = W + Cp. \quad (2)$$

At a state x , the term $W(x)$ is reward already exposed at that state; the term $\sum_y C(x, y)p(y)$ is the reward still represented by future continuation.

For an acceptance-probability problem, begin with a matrix S and an *acceptance indicator* e that is one at accepting final states and zero elsewhere. Final states have no outgoing transitions. Then

$$C_0 = S, \quad W_0 = e, \quad p_0 = (I - S)^{-1}e,$$

and $0 \leq p_0 \leq 1$: at most one accepting terminal reward can be collected. More generally, all we need below is an initial identity with a target vector in $[0, 1]^V$. An arbitrary nonnegative reward vector need not have this property, so we assume it explicitly.

A small example

Consider the graph in Figure 1. State t accepts, and state r rejects. The reward vector has $W(t) = 1$ and is zero at the other states.

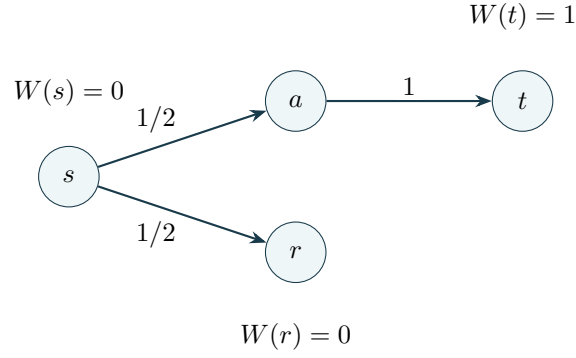


Figure 1: An acceptance probability represented by a terminal reward. Here $p(t) = p(a) = 1$, $p(r) = 0$, and $p(s) = 1/2$.

At s , equation (2) reads $p(s) = 0 + \frac{1}{2}p(a) + \frac{1}{2}p(r) = \frac{1}{2}$. At t , it reads $p(t) = W(t) = 1$. The algorithm will change the representation of these values. For example, it can eventually move the value $1/2$ into the reward at s , leaving very little continuation from s .

What would constitute progress?

Suppose we have another identity $p = W + Cp$ with $C \geq 0$ and $0 \leq p \leq 1$. Then

$$0 \leq p(x) - W(x) = (Cp)(x) \leq \sum_y C(x, y). \quad (3)$$

Thus a small row sum makes $W(x)$ a good lower approximation to $p(x)$.

The construction controls row sums indirectly. Let

$$\mathcal{A}(C) = \{v : v \text{ is the source or destination of a positive entry of } C\}$$

be the set of *active vertices*. An *entry scale* f is an upper bound on every entry of C . With this bound,

$$\sum_y C(x, y) \leq |\mathcal{A}(C)|f. \quad (4)$$

This is the *progress measure* to remember: *active count times entry scale*.

A vertex with positive reward and no incident transition is not active in this definition. This is harmless: its continuation is already zero. Active vertices measure the size of the remaining transition problem, not the support of the reward vector.

Recap

We have represented an acceptance probability by the identity $p = W + Cp$. The target p is fixed, the reward W is the part already exposed, and the continuation matrix C carries the remainder. The new quantities are the entry scale f and the active set $\mathcal{A}(C)$. Making $|\mathcal{A}(C)|f$ small will make W accurate.

2 The correction identity and Richardson's equation

Objective

Derive the exact update $(C, W) \mapsto (D, W')$, interpret it as selective expansion of continuations, and identify the constraint that makes it useful probabilistically.

One identity, valid for every correction matrix

Choose a matrix E of the same dimensions as C , and define

$$W' = (I + E)W, \quad D = C - E + EC. \quad (5)$$

The matrix E is the *correction*. The matrix D is the *residual continuation matrix*: it replaces C after this correction, on the same vertex set. The vector W' is the new exposed reward. Thus the intended new representation is $p = W' + Dp$; the matrix D carries whatever contribution has not yet moved into W' . Direct substitution gives

$$\begin{aligned} W' + Dp &= W + EW + Cp - Ep + ECp \\ &= W + Cp + E(W - p + Cp) \\ &= p. \end{aligned}$$

Equivalently,

$$I - D = (I + E)(I - C). \quad (6)$$

This is an exact algebraic identity. There is no approximation error in moving from $p = W + Cp$ to $p = W' + Dp$.

The algorithmic question is how to choose E . We want $E \geq 0$, so that $W' \geq W$. We also want $D \geq 0$, so that the remaining contribution Dp is nonnegative. Once these hold, the reward remains a lower approximation to p .

Expanding one continuation

Suppose E has just one nonzero entry, $E(x, y) = \alpha$, with $0 \leq \alpha \leq C(x, y)$. The update removes α units of the edge from x to y , adds $\alpha W(y)$ to the reward at x , and replaces the removed part by transitions of weights $\alpha C(y, z)$ to successors z of y .

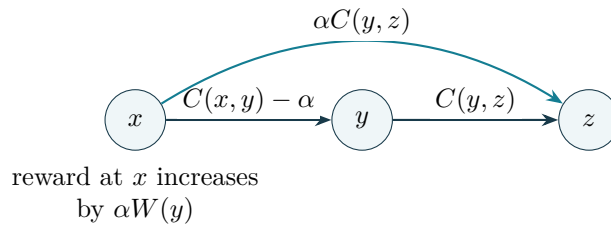


Figure 2: A correction expands part of the continuation through y . The shortcut accounts for continuing past y ; the reward update accounts for stopping to collect $W(y)$.

For a general matrix E , the term EC performs these expansions simultaneously. Later, E can have positive entries where C has none. It is therefore better to regard E as an algebraic correction built from paths, rather than always as a submatrix of the original edges.

Why this is a Richardson equation

Write $A = I - C$ and $M = I + E$. Solving $Ap = W$ by a *preconditioned Richardson step* means applying

$$z_{t+1} = z_t + M(W - Az_t).$$

Using (6), this becomes

$$z_{t+1} = MW + (I - MA)z_t = W' + Dz_t.$$

Thus D is exactly the *error-propagation matrix* of this Richardson step. Starting at $z_0 = 0$, the first iterate is $z_1 = W'$ and the error is $p - z_1 = Dp$.

The construction will repeatedly change the representation, choosing a new correction at each stage. This is not an assertion that one fixed preconditioner M is used throughout. The connection is the guiding equation and the interpretation of D as the remaining error operator.

Two useful sanity checks

First, taking $E = C$ gives

$$D = C^2, \quad W' = W + CW.$$

This is ordinary expansion by another step. It preserves nonnegativity, but it does not guarantee a decrease in the largest entry: the square of a deterministic transition can still have an entry equal to one.

Second, for every nonnegative E , whether or not $E \leq C$,

$$D\mathbf{1} = C\mathbf{1} - E(\mathbf{1} - C\mathbf{1}) \leq C\mathbf{1} \leq \mathbf{1}. \quad (7)$$

So the row-sum upper bound is automatic. The real issue is *entrywise nonnegativity* of D . A matrix can have row sums at most one while containing negative entries. Our next step is to choose E so that this cannot happen.

Recap

The correction E changes the representation by $W' = (I + E)W$ and $D = C - E + EC$, preserving p exactly. The new reward grows when $E \geq 0$; the remaining error is Dp when $D \geq 0$. Richardson's equation identifies D as an error-propagation matrix. We still need a correction rule that guarantees positivity and reduces entries.

3 A threshold rule that preserves positivity

Objective

Construct E by an explicit iteration. Understand the roles of the entry scale f , the threshold θf , and the column gate g ; prove that every finite correction has a nonnegative residual.

The scale and the cutoff

Assume $C(x, y) \leq f$ for all entries, where $0 < f \leq 1$. The *entry scale* f is a bound supplied to the stage, not a quantity recomputed as the exact maximum after every internal operation.

Fix a small constant $\theta > 0$, and use the *cutoff*

$$Q(t) = (t - \theta)_+ = \max\{t - \theta, 0\}.$$

Thus $fQ(a/f) = (a - \theta f)_+$. Subtracting this correction from a nonnegative entry a leaves

$$a - (a - \theta f)_+ = \min\{a, \theta f\}.$$

The threshold removes the excess above θf and leaves a *positive reserve*. This reserve will eventually bound the number of corrected positions in a row.

The paper uses a smoother cutoff, squeezed between two nearby shifted positive-part functions. Our hard cutoff retains the exact matrix argument. The additional regularity of the smooth cutoff matters when its inputs are only estimated; Section 10 explains the distinction.

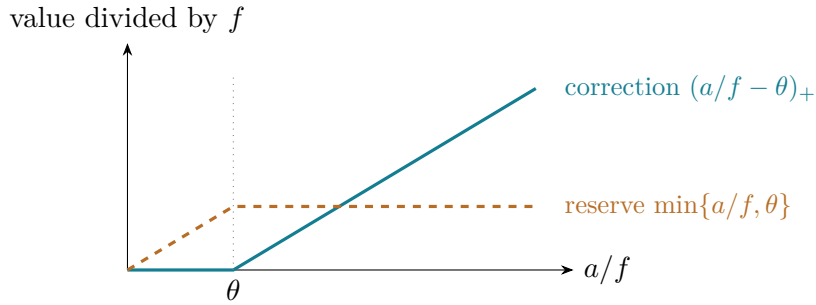


Figure 3: The cutoff separates an entry into correction and reserve. The drawing uses an enlarged threshold for visibility.

Fix the columns before iterating

Let $g : V \rightarrow [0, 1]$ be a fixed *column gate*. The same number $g(y)$ multiplies every entry of correction column y . An *open gate* $g(y) = 1$ makes the column *fully enabled*; a *closed gate* $g(y) = 0$ makes it *disabled*. Values between zero and one *partially enable* correction. Our simplified detector will use only zero and one, but the next proofs allow the entire interval $[0, 1]$.

At this point g is supplied to the iteration. It is a control choice, not an estimate of the probability vector p . In Section 5 we will simply choose $g \equiv 1$. In Section 7 a preliminary computation will choose g from the current matrix C . In either case, g stays fixed during all internal correction iterations of one stage. A new outer stage may choose a new gate.

Starting with $E_0 = 0$, form

$$G_i = C + E_{i-1}C, \quad E_i(x, y) = (G_i(x, y) - \theta f)_+ g(y) \quad (i \geq 1). \quad (8)$$

The *correction input* G_i combines the original continuation with the expansion from the preceding correction. We then correct its large entries, subject to g . Both C and f remain fixed throughout this inner iteration.

It is useful to name the *update map* now:

$$\mathcal{F}_g(E)(x, y) = ((C + EC)(x, y) - \theta f)_+ g(y), \quad E_i = \mathcal{F}_g(E_{i-1}). \quad (9)$$

A *fixed point* of this map is a correction E^* unchanged by another update: $\mathcal{F}_g(E^*) = E^*$. We will use this object in the proof of Section 4; the algorithm itself will compute only a finite number of iterates.

The scale is still f even when G_i has entries greater than f . Indeed, the simple bounds are

$$G_i \mathbf{1} \leq i \mathbf{1}, \quad E_i \mathbf{1} \leq i \mathbf{1}, \quad G_i(x, y), E_i(x, y) \leq i f. \quad (10)$$

For example, $G_i \mathbf{1} \leq \mathbf{1} + E_{i-1} \mathbf{1}$, and $G_i(x, y) \leq f + f \sum_z E_{i-1}(x, z) \leq i f$. These prove (10) by induction.

Positivity holds before the iteration has converged

Because g is fixed and Q is nondecreasing, the update in (8) is *monotone* as a function of E_{i-1} . Starting from zero therefore gives

$$0 \leq E_0 \leq E_1 \leq E_2 \leq \dots$$

After any positive number k of iterations, define $E = E_k$ and $D = C - E + EC$. Then

$$D = \underbrace{G_k - E_k}_{\geq 0} + \underbrace{(E_k - E_{k-1})C}_{\geq 0}. \quad (11)$$

The first term is nonnegative because $E_k \leq G_k$; the second is nonnegative by monotonicity. This proves $D \geq 0$ without taking a limit. Equation (7) now shows that D is substochastic. All matrices remain strictly forward because sums and products of strictly forward nonnegative matrices have no backward entries.

There is also a bound useful later:

$$D(x, y) \leq (C + E_k C)(x, y) \leq (k + 1)f. \quad (12)$$

This bound is uniform over all choices of g . It is not yet the desired decrease to f/q .

A worked three-state calculation

Take $a \rightarrow b \rightarrow c$, with both edge weights equal to one, and reward $W(c) = 1$. Here $f = 1$, $p = (1, 1, 1)$, and $W = (0, 0, 1)$. For this example only, choose $\theta = 1/4$ and $g \equiv 1$.

The first correction has $E_1(a, b) = E_1(b, c) = 3/4$. The next input has $G_2(a, c) = 3/4$, so $E_2(a, c) = 1/2$. The other two correction entries stay $3/4$. In fact, E_2 is already the fixed point on this graph.

Correction	$E(a, b)$	$E(a, c)$	$(D(a, b), D(a, c))$	$W'(a)$
E_0	0	0	(1, 0)	0
E_1	3/4	0	(1/4, 3/4)	0
E_2	3/4	1/2	(1/4, 1/4)	1/2

Each row of this table applies its indicated correction to the *same initial* pair (C, W) . These are internal correction iterations, not successive outer stages.

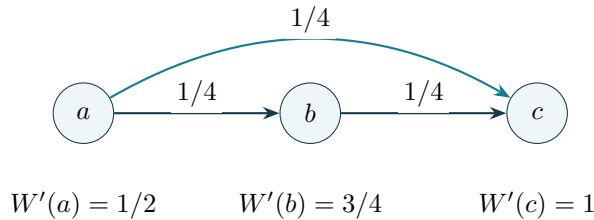


Figure 4: After correction E_2 , the value at a is $1/2 + (1/4)p(b) + (1/4)p(c) = 1$. Notice that $E_2(a, c) > C(a, c) = 0$: the correction is not restricted to the original edges.

Recap

The new players are the threshold θf , the fixed column gate g , the inputs G_i , and the corrections E_i . Monotonicity of E_i proves positivity of every finite residual. The remaining task is quantitative: show that a constant number of iterations makes entries small in every fully enabled column.

4 Why a constant number of iterations suffices**Objective**

Prove the factorial truncation bound. The essential point is that paths used to analyze one correction row all stay in one small, ordered support set.

A fixed point used only for the proof

Keep the column gate g fixed and use the update map $\mathcal{F}_g$ from (9). We seek a correction E^* that is unchanged by this update, so $E^* = \mathcal{F}_g(E^*)$. Written entrywise, this fixed-point equation is

$$E^*(x, y) = ((C + E^*C)(x, y) - \theta f)_+ g(y). \quad (13)$$

For a fixed source x , the entry $(E^*C)(x, y)$ involves only $E^*(x, z)$ with $z < y$. We can therefore determine the row in increasing order of its columns. This gives a unique strictly forward nonnegative solution E^* .

We do not need a contraction mapping theorem. The acyclic ordering solves the equation by substitution. Monotonicity of the update gives $E_i \leq E^*$ for all i .

The fixed point depends on the chosen gate: a more explicit notation would be E_g^* . It also depends on C, f, θ , which remain fixed in this discussion. For instance, $g \equiv 0$ gives $E_g^* = 0$. We suppress these subscripts to keep formulas readable. The quantitative bound below will hold uniformly for all g , which is why we can choose the iteration count before deciding which gates to close.

Define the residual obtained using this fixed-point correction:

$$T = C - E^* + E^*C.$$

This is the meaning of *fixed-point residual*: T is the residual associated with E^* . We are not claiming that T itself is a fixed point. Just as for the finite correction, the value identity is

$$p = (I + E^*)W + Tp.$$

The matrices E^* and T are *comparison objects* in the proof; the actual algorithm uses E_k and $D = C - E_k + E_kC$. Writing $a = (C + E^*C)(x, y)$, we have $E^*(x, y) \leq (a - \theta f)_+$. Thus $T \geq 0$, and $T(x, y) \geq \theta f$ whenever $E^*(x, y) > 0$. As before,

$$T\mathbf{1} = C\mathbf{1} - E^*(\mathbf{1} - C\mathbf{1}) \leq \mathbf{1}.$$

A positive reserve bounds the support of a row

Fix the source x and define its *correction support*

$$A_x = \{y : E^*(x, y) > 0\}, \quad m_x = |A_x|.$$

Every member of A_x contributes at least θf to the residual row $T(x, \cdot)$, whose total mass is at most one. Hence

$$m_x \leq \frac{1}{\theta f}. \quad (14)$$

This bound is the reason for leaving a reserve when applying the cutoff. It is a mathematical property of the exact matrix; we do not need to use a sparse data structure to exploit it in a proof.

The ordered-path count

The inequality $E^* \leq C + E^*C$ can be repeatedly substituted into itself. To see what stays fixed, write the first two substitutions:

$$\begin{aligned} E^*(x, y) &\leq C(x, y) + \sum_z E^*(x, z)C(z, y), \\ E^*(x, z) &\leq C(x, z) + \sum_w E^*(x, w)C(w, z). \end{aligned}$$

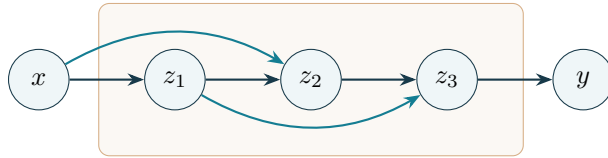
We run the construction for every source. Here we analyze one row x , and the displayed formulas show something more specific than merely fixing x for the discussion: every correction entry expanded recursively is still an entry of *row* x . We encounter $E^*(x, z)$ and then $E^*(x, w)$, rather than switching to row z or row w . Each nonzero terminal therefore belongs to the same support set A_x . The transition factors $C(w, z)$ and $C(z, y)$ connect those terminals into a path.

A resulting term is the weight of a path

$$x < z_1 < \cdots < z_j < y, \quad z_1, \dots, z_j \in A_x.$$

There are at most $\binom{m_x}{j}$ choices. Once the intermediate vertices are chosen, their order is forced. Each path weight is at most f^{j+1} . Therefore

$$E^*(x, y) \leq f \sum_{j=0}^{m_x} \binom{m_x}{j} f^j = f(1 + f)^{m_x} \leq f e^{1/\theta}. \quad (15)$$



all intermediate vertices lie in the same set A_x

Figure 5: When correction entries are recursively expanded, only entries from row x of E^* occur. Every intermediate vertex therefore lies in the same set A_x . We perform this analysis separately for every source x . Strict ordering supplies the factorial saving.

The truncation estimate

Let $\Delta_i = E^* - E_i \geq 0$ be the *truncation error* after i inner iterations. The function $t \mapsto (t - \theta f)_+$ is 1-Lipschitz, and $g \leq 1$, so

$$\Delta_i(x, y) \leq \sum_z \Delta_{i-1}(x, z)C(z, y).$$

Every nonzero $\Delta_i(x, z)$ has $z \in A_x$. Unrolling k times gives at most $\binom{m_x}{k}$ increasing sequences, each with k transition factors at most f , followed by an initial difference at most $fe^{1/\theta}$ from (15). Consequently

$$\boxed{\frac{\Delta_k(x, y)}{f} \leq e^{1/\theta} \binom{m_x}{k} f^k \leq e^{1/\theta} \frac{(1/\theta)^k}{k!}}. \quad (16)$$

If $k > m_x$, the binomial coefficient is zero. The last bound is weaker but has the great advantage of being independent of f , $|V|$, and g .

The cancellation is worth spelling out. There are two sources of scale dependence: as f gets smaller, the support bound $m_x \leq 1/(\theta f)$ gets larger, but the weight of a path with k transition factors gets smaller. Explicitly,

$$\binom{m_x}{k} f^k \leq \frac{m_x^k f^k}{k!} = \frac{(m_x f)^k}{k!} \leq \frac{(1/\theta)^k}{k!}.$$

The support choices contribute a factor of order f^{-k} , and the transition weights contribute f^k . Their cancellation removes the dependence on the current scale f .

The factorial does a separate, essential job. An unordered choice of k distinct intermediate vertices admits only one increasing order, because all transitions go forward. We therefore count subsets, giving $\binom{m_x}{k}$, rather than all ordered k -tuples, which would give m_x^k . Without this saving, the estimate after cancellation would contain $(1/\theta)^k$, which grows with k . The denominator $k!$ is what makes the bound tend to zero. Thus the scale cancellation gives uniformity across stages, while the ordering gives decay with the correction depth.

Choosing the constants once

Fix an integer *shrinkage factor* $q \geq 4$. We may take $q = 16$, matching the smallest 2^H allowed in the manuscript. Set

$$\theta = \frac{1}{64q}, \quad \text{choose a fixed integer } k \geq 1 \text{ such that } e^{1/\theta} \frac{(1/\theta)^k}{k!} \leq \theta. \quad (17)$$

This is our choice of the *inner iteration count* k . Such a k exists because the factorial eventually dominates every fixed-base exponential. No useful small numerical value of k is claimed here; it is an absolute constant for the construction.

For the finite correction $E = E_k$, write $\Delta = E^* - E$. Then

$$D = C - E + EC = T + \Delta - \Delta C \leq T + \Delta.$$

If $g(y) = 1$, equation (13) gives $T(x, y) = \min\{(C + E^*C)(x, y), \theta f\} \leq \theta f$. By (16) and (17),

$$\boxed{g(y) = 1 \implies D(x, y) \leq 2\theta f = \frac{f}{32q} < \frac{f}{q}}. \quad (18)$$

This is the progress we wanted: in every fully enabled column we obtain at least a *factor- q decrease in the entry bound*, from f to at most f/q , after a constant number k of updates. Here “factor- q ” concerns individual entries, not a claimed factor- q decrease in row sums at this point. Together with positivity and the rough bound (12) for the remaining columns, this is the complete fixed-gate lemma we will use. The corresponding argument in the original manuscript is Lemma 4.1, with a smooth cutoff.

Notation checkpoint before the algorithm

Object	Definition or identity	Meaning
C	$p = W + Cp$	Continuation before this stage's correction.
E	$E = E_k$	Finite correction computed by the algorithm.
D	$D = C - E + EC$	Continuation after correction, before any copying.
T	$T = C - E^* + E^*C$	Residual using the fixed point; a proof object.
W'	$W' = W + EW$	Reward exposed after correction.
C^+	$C^+ = D$ if there is no copying	Continuation passed to the next stage; later it will include copying.

The matrix D is an operator. The vector $Dp = p - W'$ is the actual remaining approximation error.

Recap

The proof introduced the fixed point E^* , its residual T , the row support A_x , and the error Δ_k . A reserve of θf bounds $|A_x|$; ordered paths inside that same set produce the factorial $k!$. We fixed q , θ , and a constant iteration count k . After k iterations, every fully enabled column of the finite residual has entries below f/q .

5 A complete dense algorithm with no detector

Objective

Use $g \equiv 1$ to obtain a complete algorithm already. This separates the basic correction mechanism from the additional column-control requirements that motivate the paper's detector and copies.

Enable every column

Nothing in the preceding proof prevents choosing $g(y) = 1$ for every vertex y . In this case (18) applies to every residual entry. We therefore have the following stage operation on the unchanged vertex set V .

Dense correction stage. Input: $C \geq 0$ strictly forward and substochastic, $C(x, y) \leq f$, and a reward vector W .

1. Set $E_0 = 0$. For $i = 1, \dots, k$, compute the full matrices

$$G_i = C + E_{i-1}C, \quad E_i = (G_i - \theta f)_+,$$

where the positive part is entrywise.

2. Set $E = E_k$, and output

$$C^+ = C - E + EC, \quad W^+ = (I + E)W, \quad f^+ = f/q.$$

The output matrix C^+ is nonnegative, substochastic, strictly forward, and entrywise at most f^+ . If $p = W + Cp$, then $p = W^+ + C^+p$.

The target vector p appears in the guarantee but is not an input needed to execute the stage. All computations use C and W .

Iteration and the approximation guarantee

Start with $C_0 = S$, $W_0 = e$, and $f_0 = 1$. The *outer index* ℓ counts completed *outer stages*; it is different from the *inner correction index* $i = 1, \dots, k$. Repeat the dense stage and prescribe the scale $f_\ell = q^{-\ell}$. Thus “let f be the current entry scale” means: at stage ℓ , take $f = f_\ell$, knowing inductively that $C_\ell(x, y) \leq f_\ell$. It need not be the exact largest entry. Every stage preserves the same target p_0 on the same vertex set. Hence

$$p_0 = W_\ell + C_\ell p_0, \quad 0 \leq W_\ell \leq W_{\ell+1} \leq p_0 \leq 1.$$

If $n_0 = |V|$, the maximum-entry estimate gives

$$0 \leq p_0(x) - W_\ell(x) \leq \sum_y C_\ell(x, y) \leq n_0 q^{-\ell}. \quad (19)$$

Thus $\ell \geq \lceil \log_q(n_0/\varepsilon) \rceil$ suffices for additive error at most ε when $0 < \varepsilon < 1$.

We have now completed the first story. It needed no random subsets, no sampler, no pilot, no copies, and no estimator. In particular, the detector is not necessary for the exact value-preserving correction identity or for entry decay when arbitrary full matrices are allowed.

What this result is intended to teach

An unrestricted-space algorithm could also evaluate p_0 directly by backward substitution in the acyclic graph. The point of our algorithm is the structure of its intermediate objects: a constant-depth correction at each scale, positive residuals, and a quantitative transfer of value into the reward. These are the objects the manuscript later tries to access with very little memory.

There is no new space-complexity theorem in (19). We have deliberately allowed the algorithm to retain the full matrices and exact rational numbers, with no bound here on their bit lengths. This makes the algebra visible without hiding computational costs inside an entry oracle.

Why memory brings back the detector and copies

The dense algorithm already gives the desired approximation, but its straightforward implementation stores large matrices. The remaining construction prepares it for evaluation in small space. The extra structure on E is a means to that memory goal.

For a concrete example, consider evaluating a single product entry:

$$(EC)(x, y) = \sum_z E(x, z)C(z, y).$$

If entries of the two factors were already available, we could loop over z and keep a running total. The difficulty is that those factor entries are themselves defined recursively. A suspended

computation may need to remember its endpoints x, y , its current index z , and its partial sum while evaluating a *child query*. A naive recursion keeps such information at many levels simultaneously. The issue is the accumulated *recursion stack*, even if we never materialize a full matrix.

One part of the manuscript's remedy is to control how many sources can contribute to a correction column. In suitable sampled representations, this supports recovering a source from a destination and a short label identifying the incoming edge, instead of always keeping the full source address. The bound on the full matrix alone does not prove a space bound; the sampling, access procedures, and controller are also needed.

This explains the order of the next ingredients. A *detector* tests whether a column is crowded. A preliminary *pilot* repeatedly calls this detector to choose the *column gates*, thereby controlling the correction's *incoming support*. Closing gates loses the *entry-reduction guarantee* in some columns. *Copying* repairs that loss, and a counting argument limits the number of extra *active vertices*. Section 6 begins this story. First, the optional bonus gives a different interpretation of the dense iteration we have already understood.

Recap

Setting $g \equiv 1$ gives a complete dense algorithm: the error after ℓ stages is at most $n_0 q^{-\ell}$. In particular, after $\ell = O(\log n)$ stages we get inverse-polynomial additive error when the initial state count n_0 is polynomial in the input length n . More generally the required number is $O(\log(n_0/\varepsilon))$. The new index ℓ counts outer stages, and $f_\ell = q^{-\ell}$ is their prescribed entry scale. The detector and copies enter to prepare the construction for small-space evaluation, rather than to make this dense algorithm correct.

Bonus: excess-mass transport, sandpiles, and PageRank

Objective

Give the dense correction iteration a concrete mass-transport interpretation. Identify E as cumulative emitted mass and D as remaining mass, and relate this picture to divisible sandpiles and local PageRank pushes.

One row is one mass-transport experiment

Fix one outer stage, so C, W, f remain unchanged, and take $g \equiv 1$. We now inspect the inner iterations E_i for one source x . The following notation is local to this bonus:

$$c(y) = C(x, y), \quad u_i(y) = E_i(x, y), \quad D_i = C - E_i + E_i C, \quad \mu_i(y) = D_i(x, y).$$

Think of $c(y)$ as *initial mass* placed at vertex y , $u_i(y)$ as the total mass that y has emitted by time i , and $\mu_i(y)$ as the mass currently sitting there. The *mass balance* is exactly

$$\mu_i(y) = c(y) - u_i(y) + \sum_z u_i(z) C(z, y). \quad (20)$$

The three terms are initial mass, mass sent out, and mass received from other vertices. The transport rule is fixed: when y emits an amount h , it sends $hC(y, z)$ to each z . Since $C\mathbf{1} \leq \mathbf{1}$, the unused fraction can be sent to a sink. The total mass on V never increases.

The source x specifies the initial row c . The moving mass subsequently sits at vertices $y, z, \dots$. We run one such experiment for every source x ; rows do not exchange mass with each other. The matrix E_i records all their emission histories at once.

The threshold rule says: keep the reserve, push the excess

Give every vertex *capacity* $\tau = \theta f$. At time i , a vertex whose current mass is above τ emits exactly its excess:

$$h_i(y) = (\mu_i(y) - \tau)_+.$$

All vertices do this in parallel. The proposed update is

$$u_{i+1} = u_i + h_i, \quad \mu_{i+1} = \mu_i - h_i + h_i C. \quad (21)$$

Here the last product uses row vectors. A vertex keeps $\min\{\mu_i(y), \tau\}$ of its old mass, then receives the mass emitted toward it during that round. It can therefore end the round above capacity again; a later round handles that newly arrived excess.

This is exactly our correction iteration, not just an analogy. Since $C + E_i C = E_i + D_i$, the next correction row satisfies

$$u_{i+1}(y) = (u_i(y) + \mu_i(y) - \tau)_+.$$

If $u_i(y) = 0$, subtracting $u_i(y)$ gives $(\mu_i(y) - \tau)_+$. If $u_i(y) > 0$, the finite positivity decomposition from Section 3 gives $\mu_i(y) \geq \tau$: a positive correction leaves at least the reserve. In that case the same subtraction again gives $(\mu_i(y) - \tau)_+$. Thus $u_{i+1} - u_i = h_i$. Substituting this identity into (20) proves the second update in (21).

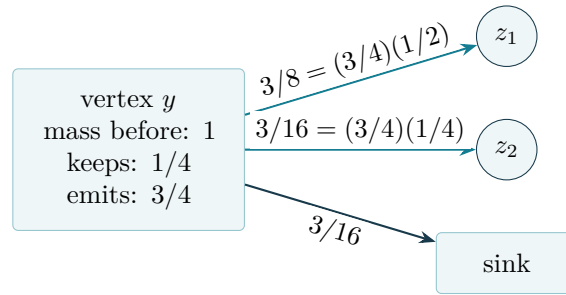


Figure 6: One local emission, using capacity $\tau = 1/4$ for illustration. The transport row has $C(y, z_1) = 1/2$, $C(y, z_2) = 1/4$, and deficit $1/4$. The emitted mass is split according to this fixed row; incoming mass from simultaneous emissions elsewhere is added afterward.

The sandpile connection explains the support bound

In the *divisible sandpile*, a vertex keeps its capacity and distributes excess divisible mass to its neighbors. The cumulative emission is called the *odometer*. Levine and Peres describe this formulation and its relation to potential theory; *parallel toppling* is also treated explicitly by Levine, Murugan, Peres, and Uğurcan. Their standard graph models distribute mass according to graph adjacency. Our formulas above give the corresponding directed, weighted, dissipative version, with transport matrix C and capacity τ .

The dictionary for our particular iteration is:

Matrix object for source x	Mass-transport meaning
$C(x, \cdot)$	Initial mass distribution, with total mass at most one.
$E_i(x, \cdot)$	Odometer: cumulative emitted mass.
$D_i(x, \cdot)$	Mass remaining after i parallel rounds.
$E^*(x, \cdot)$	Odometer at stabilization.
$T(x, \cdot)$	Stabilized mass, at most τ at every vertex.
A_x	Vertices that emit a positive amount.

Once a vertex emits, it keeps a reserve of τ . Subsequent emissions again leave that reserve, and incoming mass can only add to it. At stabilization, a vertex that has emitted therefore has exactly τ mass. Since the total mass is at most one,

$$|A_x|\tau \leq 1, \quad |A_x| \leq \frac{1}{\theta f}.$$

This recovers the support bound from Section 4. The acyclic ordering ensures that the iteration stabilizes if continued; however, the algorithm does not need to wait for exact stabilization. The factorial estimate shows that a constant number of rounds already approximates the odometer sufficiently well, uniformly in the graph size and the scale f .

For the three-state example $a \rightarrow b \rightarrow c$ from Section 3, examine source $x = a$. Initially there is mass 1 at b . With capacity $1/4$, the first round sends $3/4$ from b to c . The next round sends the excess $1/2$ from c to the sink, leaving $1/4$ at both b and c . Thus the odometer is $u(b) = 3/4, u(c) = 1/2$, exactly the row $E^*(a, \cdot)$ in our earlier table.

Where the reward goes during a push

The mass picture also explains the reward update. For our fixed source x , let

$$B_i = W(x) + \sum_y u_i(y)W(y) = ((I + E_i)W)(x).$$

This is the exposed reward after the cumulative emissions u_i . When a vertex y emits h , we add $hW(y)$ to this scalar reward and send mass $hC(y, z)$ onward. The identity

$$h p(y) = h W(y) + \sum_z h C(y, z) p(z)$$

shows exactly why the total represented value is preserved. Consequently

$$p(x) = B_i + \sum_y \mu_i(y) p(y)$$

at every round. The reward is an accounting of value; in our general model $W(y)$ need not equal the amount of transport mass lost from row y .

In the chain example, emitting $3/4$ from b adds no reward because $W(b) = 0$. Emitting $1/2$ from c adds reward $1/2$ because $W(c) = 1$. The remaining two masses of $1/4$ each carry the other half of the value. This reproduces $W'(a) = 1/2$ and $D(a, b) = D(a, c) = 1/4$.

The related PageRank viewpoint, and the flow question

The *local PageRank algorithm* of Andersen, Chung, and Lang maintains an accumulated approximation together with a nonnegative residual. A *push* at one vertex moves part of its residual into the approximation and distributes the remainder using a random-walk transition. Its invariant expresses the target PageRank vector as the accumulated vector plus the PageRank contribution of the residual. See Section 3 of their paper.

Our identity $p(x) = B_i + \mu_i p$ has the same accounting pattern: expose part of the answer, and leave the rest represented by nonnegative continuation. The update rules differ. PageRank pushes use a stopping parameter and their own threshold for choosing vertices; our exact correspondence to parallel excess toppling uses the hard reserve θf and $g \equiv 1$.

This also explains the resemblance to a flow algorithm. Mass is routed through a graph and balanced locally. But no routing choices are optimized: the proportions are prescribed by C . The capacity θf bounds mass left at a vertex after it emits its old excess; it is not an upper bound on the cumulative amount of flow through an edge.

These are mathematical connections, not a claim about historical influence on the supplied manuscript. Nor do they identify the full smooth detector-and-copy algorithm with a classical sandpile process. Their value here is to make the dense correction, its reserve, and its residual concrete.

Pointers for the connection

For the standard sandpile and odometer formulation, see Levine and Peres, *Scaling Limits for Internal Aggregation Models with Multiple Sources*, Sections 1 and 3. For parallel toppling, see Levine, Murugan, Peres, and Uğurcan, *The Divisible Sandpile at Critical Density*, Section 2. For residual pushes, see Andersen, Chung, and Lang, *Local Graph Partitioning using PageRank Vectors*, Section 3. The equivalence to our recurrence was derived directly above.

Recap

For the hard cutoff with $g \equiv 1$, each correction row is the odometer of a parallel excess-mass process. The residual row is the mass still present; the reward records the value already exposed. The reserve θf explains why only $1/(\theta f)$ vertices can emit in one experiment. This viewpoint helps interpret the dense algorithm, while the factorial argument explains why we may stop after a constant number of rounds.

6 What goes wrong with unrestricted incoming degree?

Objective

Explain the additional structural goal behind the detector. Distinguish a row-support bound from a column-support bound, and see why disabling some columns creates a role for copying.

Rows are controlled; columns need not be

The factorial proof leading to (16) used the *row-support bound* (14): each row of E^* has at most $1/(\theta f)$ nonzero entries. It did not bound the number of sources that can point to a fixed

destination y . A substochastic matrix has bounded outgoing mass per source, but arbitrarily many different sources can concentrate mass in one column.

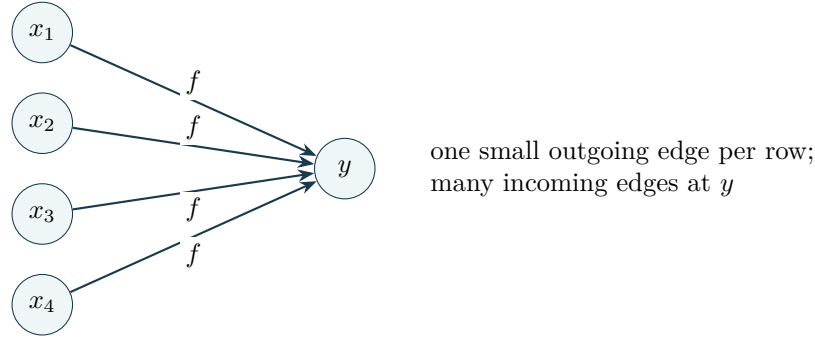


Figure 7: A column can have arbitrarily many substantial entries, even when each row is very simple. The four displayed sources can be replaced by any number of sources.

For this star, the first fully enabled correction has $E_1(x_j, y) = (1 - \theta)f$ for every source. There is no bound of the form R/f on the column support, with R independent of the number of sources.

This example concerns a general intermediate matrix. Even if the original branching program has bounded incoming degree, later shortcut matrices need not inherit it. Likewise, bistochastic one-step transitions do not automatically make every selective correction column-balanced.

The extra property we will enforce

We now ask that every correction column contain at most

$$\#\{x : E_i(x, y) > 0\} \leq R/f \quad (22)$$

nonzero entries, for a fixed constant R . We call this *incoming-degree control at scale f* . The bound still depends on f ; it is not a constant-degree claim about the full matrix.

A preview: why a sampling rate of f fits this bound

Here “later” refers to the full manuscript’s small-space implementation. It represents matrix operations using *sparse, sample-dependent tables*, with *normalization identities* linking their entries back to the full matrices. The purpose is to support local access and recursive evaluation with little memory. We will not construct those tables in these notes; the original manuscript defines its sampled vertex sets in Section 3 and the matrix representations in Section 5.

To *select* a source means to include it in a sampled subset S of the source vertices. Let $X_f(x) = 1$ mean that $x \in S$. If each source is selected with probability proportional to f , then the number of selected sources entering a fixed correction column y has expectation

$$\mathbb{E} \left[\sum_{x: E_i(x, y) > 0} X_f(x) \right] \leq \frac{R}{f} O(f) = O(R).$$

No independence is needed for this expectation calculation. For example, among 100 possible incoming sources, selecting each with probability $1/100$ gives one selected source on average. A particular sample can contain zero, one, or several of them. The choice of a source is shared across destinations: this describes selected source rows, not independent choices of every entry.

The full matrix is still the mathematical object being represented; a sample is one view used in its implementation. The estimate above explains the matching scales R/f and f , but an expectation bound is not a deterministic degree bound. The manuscript adds gates and normalization to obtain its actual access guarantees. In Section 7 we will only prove the deterministic full-matrix bound (22), without sampling.

Which columns may we enable?

Suppose we *enable* correction in column y , meaning that we choose $g(y) = 1$. By the correction rule (8),

$$E_i(x, y) = \max\{G_i(x, y) - \theta f, 0\}, \quad E_i(x, y) > 0 \iff G_i(x, y) > \theta f.$$

The equivalence is immediate from the definition: equality at the threshold gives zero. Enabling a column is a choice we make, not a statement that the column already satisfies a support bound. Recall that the dense algorithm enabled every column.

We now want to make this choice only where at most R/f sources exceed the threshold at each correction step. This is a bound on the number of substantial entries, rather than on the total incoming mass. It is the structural requirement that the dense algorithm did not impose.

The *detector* is an algorithm that tests a supplied matrix for crowded columns. In our version it simply counts entries above a lower threshold δf , where $0 < \delta < \theta$, and reports a column as *overloaded* if that count exceeds R/f . This is a very simple detector; in the original manuscript the same role is implemented by a more elaborate sampled construction, designed for the later small-space algorithm.

The *pilot* is the preliminary pass that repeatedly invokes this detector on its evolving input matrices and remembers all flags. Its final flagged set determines g . Thus the detector is a subroutine of the pilot; it is not another pass performed after the pilot. Section 7 will make both routines explicit and prove the support bound for the actual correction they produce.

Into a column, then through its vertex

There are two viewpoints on a correction column y . The entries $E(x, y)$, as x varies, form column y of the correction matrix. When multiplied by C , the same entries specify expansions *through* the intermediate vertex y :

$$E(x, y)p(y) = E(x, y)W(y) + \sum_z E(x, y)C(y, z)p(z).$$

The correction subtracts the left-hand contribution from the old continuation into y , exposes its reward, and replaces the remaining part by continuations to later vertices z .

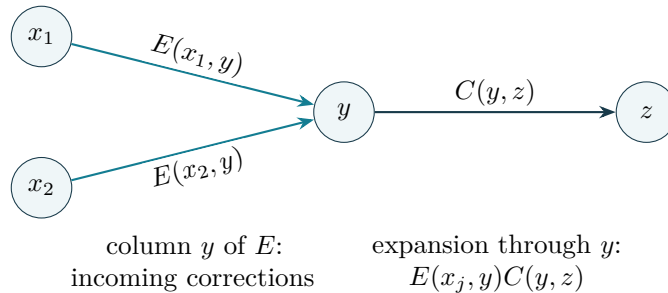


Figure 8: Vertex y is a destination in E and an intermediate vertex in EC . Closing $g(y)$ removes the displayed correction entries and their expansions to z . It does not delete the residual continuation into y . The arrows labeled E represent correction amounts, not necessarily original edges of C .

This explains why a crowded correction column can affect later columns: each of its source entries may expand along outgoing transitions from y . The bound we impose directly is the number of sources with $E(x, y) > 0$, which also serves the incoming-access requirements of the later implementation.

What closing a gate does, and what it costs

For a column y flagged by the pilot, set $g(y) = 0$ before running the actual correction. Then $E_i(x, y) = 0$ for every x, i . In the expansion viewpoint, we have disabled correction *through* y . To see its effect on entries *into* y , write the residual formula entrywise:

$$D(x, y) = C(x, y) + \sum_z E(x, z)C(z, y) - E(x, y).$$

The sum records expansions through earlier intermediate vertices z that arrive at y . The subtraction $-E(x, y)$ is the amount we in turn expand through y . Closing the gate removes that subtraction, leaving

$$D(x, y) = C(x, y) + (EC)(x, y).$$

These entries into y may remain large or even increase. Thus disabling expansion *through* y sacrifices the entry-reduction guarantee *in column* y of D . The contribution $D(x, y)p(y)$ still represents the appropriate future value; we have not prevented the underlying process from continuing beyond y .

Closing this gate also changes contributions to later columns, but their own entry-reduction guarantees remain valid whenever their gates are fully open, by (18). Correctness and positivity hold for any fixed gate choice used throughout the actual iteration. If we close every gate, $E = 0$, $W' = W$, and $D = C$: the representation remains correct but makes no progress.

The division of labor, made explicit

We have two requirements: all entries of the next continuation matrix must be at most f/q , and each correction column must have support at most R/f . The operations have different responsibilities.

1. **Use the pilot to choose the gates.** By definition, a column never flagged by our pilot passes all its detector count tests. Section 7 will prove that these bounds transfer from pilot inputs to the actual correction, giving incoming support at most R/f . A flagged column has gate zero, so its actual correction support is empty.
2. **Correct the enabled columns.** With $g(y) = 1$, the finite threshold iteration gives $D(x, y) \leq f/q$ for every x , by (18). These residual entries already satisfy the desired bound, so we leave them unchanged during the copying step.
3. **Copy destinations of remaining large entries.** By *splitting an entry* of weight d into y , we mean replacing it by weights d/m into m copies of y , each carrying the same continuation value. The total contribution stays $dp(y)$. The rough bound (12) is $D(x, y) \leq Kf$, where $K = k + 1$, so $m \geq Kq$ suffices to make each piece at most f/q . Section 8 defines the full copied transition system.

Copying addresses entry size; it does not promise to reduce the number of sources entering each copy. The incoming-support condition is imposed on the correction E , whereas the entry bound is imposed on the next continuation matrix C^+ .

There is a price for copying: more vertices. Copying alone merely redistributes continuation mass. If we divided every entry among m copies of every destination, the maximum entry could fall by

m while the active count grew by m , giving no improvement in their product. We instead need the overloaded destinations to be rare. The next section proves this by counting witnesses: every flagged column needs many substantial incoming entries, while the total outgoing mass of each row is bounded. This count will pay for enough copies to repair the exceptional columns, while increasing the active count by at most two.

The remaining dependency is how to choose the gates. The pilot examines its successive inputs, permanently flags overloaded columns, freezes the final gate g , and then restarts the actual correction from zero. Restarting is what allows us to use the fixed-gate positivity and factorial proofs without alteration.

Recap

Row control and column control are different. We now want the correction's incoming support to be at most R/f per column. The new ideas are overload detection, permanent flags, and a pilot that chooses a fixed column gate g . Disabled columns may have large entries requiring copying. The pilot's detector tests will control their number, and a comparison proof will transfer the pilot's count bounds to the actual correction. An incidence count will ensure that only a small set of parents needs extra active copies.

7 A direct-count detector and its pilot

Objective

Give an explicit deterministic detector, prove both incoming-degree control and a bound on the number of flagged columns, and explain why the actual iteration is restarted after the pilot.

Who calls whom, and what each routine returns

The outer algorithm runs for $L = O(\log n)$ stages when the initial graph has polynomial size and the desired error is inverse-polynomial. Each stage runs a *pilot pass* of k iterations. In each pilot iteration, the pilot calls the *detector* on its current input matrix. After the pilot finishes, an *actual correction pass* runs for k iterations with the final gates fixed. This second pass does not call the detector.

Routine	Input	Output or responsibility
Detector	One matrix G and scale f	Columns with too many entries above δf .
Pilot	The stage's C, f	Repeatedly calls the detector; accumulates flags O and returns $g = \mathbf{1}_{V \setminus O}$.
Actual correction	The same C, f , and fixed g	Computes E_k to use in the reward and residual update.
Outer algorithm	Initial S, e and target accuracy	Repeats complete stages, changing C, W, f between them.

The detector is just a count-and-compare algorithm here. Its more elaborate sampled counterpart in the original manuscript performs the same overload-testing role. Figure 12 in Section 9 shows the complete call and loop structure.

Why choosing the gates requires a preliminary pass

Checking only C would not suffice: the relevant inputs are $G_i = C + E_{i-1}C$, and the shortcut terms can create substantial entries absent from C . A column can therefore become crowded at a later inner iteration.

The more delicate issue is that the gates themselves affect which columns become crowded. For example, opening the gate at y permits $E_{i-1}(x, y) > 0$, which can contribute $E_{i-1}(x, y)C(y, z)$ to a later input entry $G_i(x, z)$. Closing that gate suppresses this contribution. Thus the dependence runs in the direction

$$\text{chosen gates} \longrightarrow \text{corrections} \longrightarrow \text{crowding in later inputs.}$$

Yet we want the observed crowding to help us choose the gates. We could certainly select some gates in advance and run the iteration, but we would not thereby know that the resulting correction has the required incoming support. We need a procedure that makes the choices and establishes the bound.

The pilot makes the choices progressively. Start with zero correction and no flags. Form the first input, call the detector, and close newly flagged columns before computing the first pilot correction. Use that correction to form the next input, test again, and continue. Once a column is flagged, keep its gate closed for the rest of this pass. After k iterations, every decision has been made and the final fixed gate g is known.

This removes the dependency from the definition of the algorithm: every pilot operation uses an already computed matrix. But the pilot has paid a price. Its gates change while it runs, so its correction sequence need not be monotone. We will therefore keep the pilot's *decisions* and then recompute a correction under those fixed decisions. The reason this restart restores the proof's invariants is spelled out below, after the equations.

Recall the constants and the two iteration indices

Keep q, θ, k as chosen in (17). In particular, k makes

$$e^{1/\theta} \frac{(1/\theta)^k}{k!} \leq \theta,$$

so the factorial bound (16) gives $E_g^* - E_k \leq \theta f$ entrywise. This choice is uniform over every fixed gate $g : V \rightarrow [0, 1]$. We can therefore choose k *before* the pilot determines g , and use the same k for the pilot and the actual pass. It is a fixed constant, not the $O(\log n)$ number of outer stages.

Set the *detector threshold parameter* to $\delta = \theta/2$. Let $R > 0$ be the constant in the *incoming-support budget* R/f ; its size will be fixed in (30), after choosing the number of copies.

The following formulas describe one outer stage with input C, f . More explicitly, at stage ℓ they use $C = C_\ell$ and $f = f_\ell = q^{-\ell}$. We suppress ℓ here to focus on the pilot index $i = 1, \dots, k$. The pilot uses the same code at every outer stage, but its input matrix, scale, and possibly vertex set change. Each new stage starts a fresh pilot with zero correction and no flags.

Within one pilot run, C and f stay fixed, while the pilot correction and flagged set evolve. Neither this pass nor the actual correction pass updates the reward W . Both prepare a single correction E for this stage; only afterward do we compute $D = C - E + EC$ and $W' = W + EW$.

The detector inside the pilot equations

Hats distinguish *pilot objects* $\hat{E}_i, \hat{G}_i$ from the actual correction objects E_i, G_i . Every object here is computed exactly. A hat does not mean an approximation or a noisy estimate. Let $\hat{E}_i$ be the pilot correction after step i , $\hat{G}_i$ its input, $a_i(y)$ the *column count*, and O_i the *flagged set* accumulated through that step. The *pilot gate* $\hat{g}_i$ may change with i .

Initialize $\hat{E}_0 = 0$ and $O_0 = \emptyset$. For $i = 1, \dots, k$, compute

$$\hat{G}_i = C + \hat{E}_{i-1}C, \quad (23)$$

$$a_i(y) = \#\{x : \hat{G}_i(x, y) > \delta f\}, \quad (24)$$

$$O_i = O_{i-1} \cup \underbrace{\{y : a_i(y) > R/f\}}_{\text{detector's output on } \hat{G}_i}, \quad (25)$$

$$\hat{g}_i(y) = \mathbf{1}_{\{y \notin O_i\}}, \quad \hat{E}_i(x, y) = (\hat{G}_i(x, y) - \theta f)_+ \hat{g}_i(y). \quad (26)$$

The detector consists of the count in (24) and the test $a_i(y) > R/f$ under the brace in (25). The union with O_{i-1} is the *pilot's memory*: it preserves earlier flags. These are distinct responsibilities within one iteration.

Read the equations in order. Form the input $\hat{G}_i$, call the detector on its columns, accumulate flags, and compute $\hat{E}_i$ using the updated gates. A column first flagged at step i is already disabled in $\hat{E}_i$. At the next iteration the detector runs the same count-and-compare code on a potentially different matrix $\hat{G}_{i+1}$.

By definition, a column unflagged throughout this pilot has $a_i(y) \leq R/f$ for every i . This is already the required count bound for the *pilot inputs*. What remains to prove is that the actual correction, computed in the separate pass, inherits it. That is the purpose of the domination argument below; there is no additional “certificate” computed by the detector.

The quantity $a_i(y)$ counts sources, rather than summing their weights. This matters because the later access problem asks how many possible predecessors must be identified. The two thresholds play different roles: δf decides whether an entry is a witness for the detector, while θf decides whether it can receive a positive correction. Since $\delta < \theta$, every positive actual correction will have a corresponding witness in the dominating pilot input.

Flags are permanent within this stage: $O_{i-1} \subseteq O_i$. A column is never reopened after being flagged, even if a later pilot input there becomes smaller. This permanence is what will let the final gate lie below every gate used in the pilot.

Every quantity in this definition is available before it is used. In particular, $\hat{G}_i$ is already a fixed matrix when a_i is computed. No random sample or estimated matrix appears in these definitions.

Freeze, then restart

After the pilot, set

$$O = O_k, \quad g(y) = \mathbf{1}_{\{y \notin O\}}.$$

Now restart from $E_0 = 0$ and run the fixed-gate iteration

$$G_i = C + E_{i-1}C, \quad E_i = (G_i - \theta f)_+ g \quad (i = 1, \dots, k). \quad (27)$$

Here and below multiplication by g is columnwise. Finally set $E = E_k$ and $D = C - E + EC$.

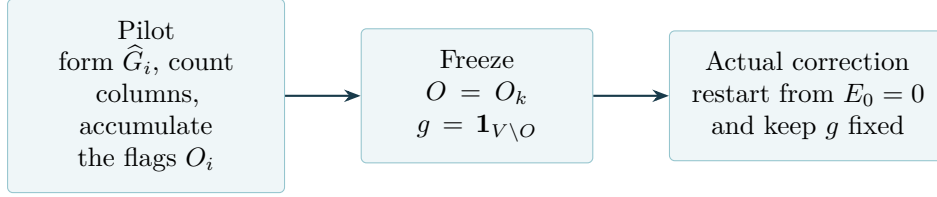


Figure 9: The pilot chooses the column gates. The actual iteration starts over with those gates fixed, giving the monotonicity needed for positivity and factorial truncation. A changing pilot gate can make a pilot correction decrease.

Why we keep the decisions but recompute the correction

Why not use the pilot's last matrix $\widehat{E}_k$ directly? The pilot was designed to discover which gates to close. If a column receives positive correction at one step and is flagged at the next, that entire correction column is set to zero. Hence we cannot conclude

$$\widehat{E}_0 \leq \widehat{E}_1 \leq \dots \leq \widehat{E}_k.$$

This matters for a specific reason. The finite positivity proof in (11) wrote

$$D = \underbrace{G_k - E_k}_{\geq 0} + \underbrace{(E_k - E_{k-1})C}_{\geq 0}.$$

The second term was nonnegative because the correction sequence was increasing. A changing pilot gate can destroy that property. The pilot's final matrix therefore does not come with the guarantee proved by this argument. We are not asserting that every pilot residual is negative; we are identifying the hypothesis that no longer holds.

Now freeze all the pilot's final decisions. A column ever flagged gets gate zero from the *beginning* of the actual pass, and every other column keeps gate one throughout. Restart from $E_0 = 0$. The same monotone update map $\mathcal{F}_g$ is applied at every step, so

$$0 = E_0 \leq E_1 \leq \dots \leq E_k.$$

The positivity proof applies again, and the fixed-gate factorial estimate gives the desired entry reduction in open columns.

The pilot's lasting output is its set of decisions O , equivalently g . Its correction values served to guide those decisions; the actual pass recomputes the correction with all decisions already in force. We do not start that pass from $\widehat{E}_k$, and we do not obtain E_k merely by erasing columns of the pilot's last matrix. Earlier gate changes affect expansions into other columns as well, so the whole recurrence is run again.

There is still one question: why should the detector's conclusions about pilot inputs tell us anything about these recomputed actual inputs? The next comparison answers exactly that question.

The pilot dominates the actual correction

We prove *domination*: at each corresponding step, the actual correction and input are entrywise at most the pilot correction and input. The actual pass imposes every eventual restriction from its first step; the pilot may discover some restrictions only later. This gives a comparison between the two passes even though the pilot sequence itself need not increase with time.

Since flags only accumulate, $g \leq \widehat{g}_i$ for every i . Induction gives

$$E_i \leq \widehat{E}_i, \quad G_i \leq \widehat{G}_i. \quad (28)$$

Here is the induction explicitly. Both corrections start at zero. If $E_{i-1} \leq \widehat{E}_{i-1}$, nonnegativity of C gives

$$G_i = C + E_{i-1}C \leq C + \widehat{E}_{i-1}C = \widehat{G}_i.$$

The cutoff is nondecreasing, and the final gate satisfies $g(y) \leq \widehat{g}_i(y)$, so

$$E_i(x, y) = (G_i(x, y) - \theta f)_+ g(y) \leq (\widehat{G}_i(x, y) - \theta f)_+ \widehat{g}_i(y) = \widehat{E}_i(x, y).$$

This closes the induction. It compares the two runs at the same index i ; it does not assume $\widehat{E}_i \geq \widehat{E}_{i-1}$.

If $y \notin O$, then $a_i(y) \leq R/f$ for every i by the definition of the pilot flags. The proof transfers that count bound to the actual correction through the implication

$$E_i(x, y) > 0 \implies G_i(x, y) > \theta f \implies \widehat{G}_i(x, y) > \delta f.$$

The second implication uses domination and $\delta < \theta$. Thus every positive actual correction entry corresponds to a source counted by the pilot detector, and it follows that

$$\#\{x : E_i(x, y) > 0\} \leq R/f.$$

If $y \in O$, its actual correction column is identically zero. Thus the incoming-degree bound holds for every column and every actual iteration.

Why few columns are flagged

Let $n' = |\mathcal{A}(C)|$. Just as in (10),

$$\widehat{G}_i \mathbf{1} \leq i \mathbf{1}, \quad \widehat{G}_i(x, y) \leq i f.$$

Rows and columns that are zero in C stay zero in all pilot inputs: a zero source row stays zero by induction, and a zero destination column stays zero in every product ending in C .

A row of $\widehat{G}_i$ can have at most $i/(\delta f)$ entries exceeding δf . There are at most n' nonzero rows. Therefore

$$\sum_{i=1}^k \#\{(x, y) : \widehat{G}_i(x, y) > \delta f\} \leq \frac{n'}{\delta f} \sum_{i=1}^k i = \frac{n'k(k+1)}{2\delta f}.$$

To understand the count, regard a witness as a *witness triple* (i, x, y) : at step i , source x contributes a substantial entry to destination y . The displayed sum bounds the total number of such triples. It is legitimate to count the same pair (x, y) again at another step, because we have explicitly budgeted for all k steps.

Each flagged column has more than R/f witnesses at its first flagging step. Choose that step once for each flagged column. Witness triples charged to different columns have different last coordinates, so none is charged twice. In other words,

$$\underbrace{|O| (R/f)}_{\text{witnesses required by flagged columns}} \leq \underbrace{\frac{n'k(k+1)}{2\delta f}}_{\text{all available witness triples}}.$$

Dividing by the per-column requirement yields

$$\boxed{|O| \leq \frac{n'k(k+1)}{2\delta R}.} \quad (29)$$

The factor $1/f$ cancels. This is crucial: a late stage has a very small entry scale, but the fraction of vertices that can be flagged is still bounded by an absolute constant depending only on our chosen parameters.

What the pilot and detector have achieved

Combining this section with the fixed-gate lemma gives three guarantees:

$$\begin{aligned} \#\{x : E_i(x, y) > 0\} &\leq R/f && \text{for every actual correction column;} \\ D(x, y) &\leq f/(32q) && \text{if } y \notin O; \\ D(x, y) &\leq (k+1)f && \text{for all } x, y. \end{aligned}$$

The first is the new structural input for later access procedures; the second says that unflagged columns already have tiny entries; the third bounds how much splitting could be needed in flagged columns. The bound on $|O|$ supplies a fourth fact: only a small fraction of destinations require that repair. These are the only outputs that the copying step needs from the detector. The analogous pilot and detector guarantees in the original manuscript are Lemmas 4.2–4.3.

Recap

The detector counts and tests one input matrix. The pilot repeatedly calls it, remembers flags O_i , and returns a fixed gate g . Because gates can change during the pilot, its corrections need not increase. We keep its decisions and restart the actual pass from zero, recovering monotonicity and positivity. Domination then transfers the pilot's count bounds to the actual correction. The new objects are $\hat{G}_i, \hat{E}_i, a_i(y), O_i$; the guarantees are incoming support at most R/f and the small flagged-set bound (29).

8 Copying the exceptional destinations

Objective

Reduce all residual entries to at most f/q , including those in flagged columns, while at most doubling the number of active vertices. Prove the projection identity that preserves values across the change of vertex set.

What a copy must accomplish

At the end of Section 7, we have the corrected identity $v = W' + Dv$ on the old vertex set V . We temporarily write the target as v because the next operation changes its indexing. The matrix D is already nonnegative and substochastic. Its unflagged columns have entries below f/q , and every entry, flagged or not, is at most $(k+1)f$.

Suppose $D(x, y) = d > f/q$. We want to replace this one large contribution $dv(y)$ by several smaller contributions without changing their sum. Splitting it among m destinations of value $v(y)$ does exactly this:

$$dv(y) = \sum_{j=0}^{m-1} (d/m) v(y).$$

This observation leaves two tasks. We must define the outgoing behavior and reward of a copy so that it really has the required value. We must also show that the number of copies does not overwhelm our progress in entry size. The factors r_i , fractions s_j , and matrix P below are bookkeeping for these two tasks.

Choose the copy budget after choosing the correction depth

Write $K = k + 1$. Choose the *copy number* to be an integer $m > Kq$ and then choose R large enough that

$$R \geq \frac{(m-1)k(k+1)}{2\delta}. \quad (30)$$

The number m pays for the worst possible entry size: $Kf/m < f/q$. The constant R has a different job. Raising R permits a larger, but still constant-scaled, incoming-support budget R/f and makes overload flags rarer. We choose it large enough that these rare flags can afford $m-1$ extra copies each.

All of these are constants. Their order of choice is

$$q \longrightarrow \theta, k \longrightarrow \delta, m \longrightarrow R.$$

There is no circular dependency: the factorial argument chose k uniformly for every possible column gate g .

By (29) and (30),

$$(m-1)|O| \leq |\mathcal{A}(C)|. \quad (31)$$

This is exactly the budget needed for the extra copies.

The copy universe and activation factors

Create a *formal copy set* $V^+ = V \times \{0, \dots, m-1\}$, and write $x_i = (x, i)$. Copy x_0 is the *primary copy*. Define *row activation factors*

$$r_0(x) = 1, \quad r_i(x) = \mathbf{1}_{\{x \in O\}} \quad (i > 0). \quad (32)$$

Thus extra copies have outgoing behavior only when their parent was flagged. The primary copy always keeps its parent's behavior. This includes old vertices with zero outgoing rows: their primary copies still keep the old reward, even though that alone does not make them active in our transition-based definition.

The formal domain contains all m labels for every parent, for uniform bookkeeping. For an unflagged parent, the nonprimary labels will have zero reward, zero outgoing row, and no incoming edges. They are placeholders, not extra active states. An activated copy of a flagged parent replicates the parent's corrected outgoing behavior and reward. Its value is not divided by m .

We also need *routing fractions* for each residual entry. If $D(x, y) \leq f/q$, route it entirely to the primary copy y_0 . If $D(x, y) > f/q$, distribute it uniformly among all m copies of y . Formally, write $s_j(x, y)$ for these fractions:

$$(s_0(x, y), \dots, s_{m-1}(x, y)) = \begin{cases} (1, 0, \dots, 0), & D(x, y) \leq f/q, \\ (1/m, \dots, 1/m), & D(x, y) > f/q. \end{cases} \quad (33)$$

The new transition matrix is

$$C^+(x_i, y_j) = r_i(x)D(x, y)s_j(x, y). \quad (34)$$

Read (34) from left to right. The source activation $r_i(x)$ asks whether copy x_i is allowed to behave like its parent. The residual entry $D(x, y)$ gives the parent's total weight toward y . The routing fraction $s_j(x, y)$ says how much of that weight is assigned to destination copy y_j .

These are two different choices. The factor $r_i(x)$ belongs to a source copy and is the same across all its destinations. The fraction $s_j(x, y)$ belongs to a particular edge and can differ for different

sources entering the same parent y . A flagged destination can have both small incoming edges, sent only to y_0 , and large incoming edges, split among all its copies.

For example, an entry $d \leq f/q$ stays intact at y_0 . An entry $d > f/q$ is replaced by m pieces d/m . In either case the outgoing row total from an activated source copy is unchanged. This is why copying by itself does not reduce remaining mass: it changes the number and sizes of entries.

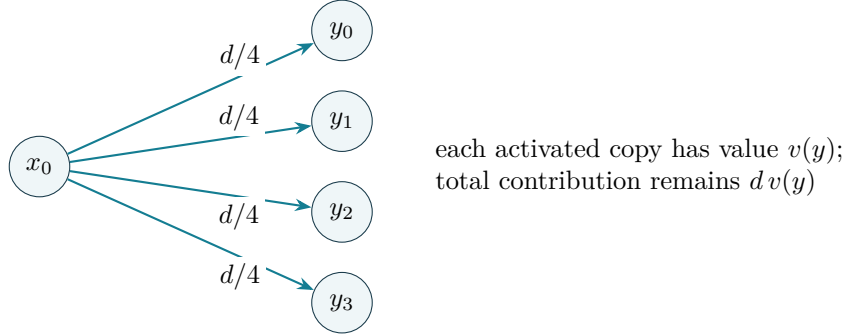


Figure 10: Splitting an edge of weight d among four activated destination copies. Four is used only to illustrate the routing rule; the theorem uses the larger constant $m > Kq$.

The entry and row-sum bounds

Since the routing fractions sum to one,

$$\sum_{y,j} C^+(x_i, y_j) = r_i(x) \sum_y D(x, y) \leq 1.$$

If an edge is not split, its weight was already at most f/q . If it is split, each piece has weight at most $Kf/m < f/q$. Multiplication by $r_i(x) \leq 1$ can only decrease it. Hence C^+ is substochastic and every entry is at most f/q .

Order copies by their parent first. Because $D(x, y) > 0$ only for $x < y$, every new edge still goes forward.

Why the values are preserved

A large residual entry $D(x, y) > f/q$ cannot belong to an unflagged column, by (18). Thus every nonprimary destination that receives a positive routing fraction has parent $y \in O$. Its activation factor is therefore one. The primary factor is always one. It follows that

$$D(x, y) > 0 \implies \sum_{j=0}^{m-1} s_j(x, y) r_j(y) = 1. \quad (35)$$

This consistency between routing and activation is the essential invariant of the copy construction. If we sent positive weight to a disabled placeholder, its value would be zero and the old contribution would be lost. The gate analysis prevents that: only flagged parents can need a split, and every copy of a flagged parent is activated.

Define the *projection matrix* P of dimensions $V^+ \times V$ by

$$P(x_i, z) = r_i(x) \mathbf{1}_{\{z=x\}}. \quad (36)$$

For a vector v on V , the *lifted vector* is $(Pv)(x_i) = r_i(x)v(x)$. Although we call P a projection because it refers a copy back to its parent, its action on value vectors goes from the old domain

to the larger domain. It copies values; it does not average or sum the values of siblings. In our binary version, an activated child gets $v(x)$ and a disabled child gets zero.

The dimensions are worth checking: D is $|V| \times |V|$, C^+ is $|V^+| \times |V^+|$, and P is $|V^+| \times |V|$. Thus both C^+P and PD map an old value vector to a new one. Equation (35) gives

$$\begin{aligned} (C^+P)(x_i, y) &= \sum_j r_i(x) D(x, y) s_j(x, y) r_j(y) \\ &= r_i(x) D(x, y) = (PD)(x_i, y). \end{aligned}$$

Therefore

$$\boxed{C^+P = PD.} \quad (37)$$

In words, lifting the values and then following the copied transitions has the same effect as following the old residual transitions and then lifting the result.

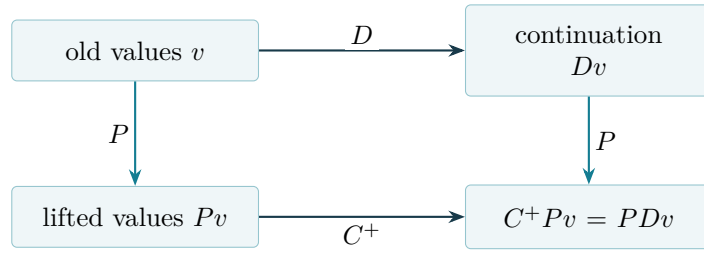


Figure 11: Two ways to transport the same continuation value. The equality of the two routes is $C^+P = PD$. The vertical arrows duplicate or zero values according to activation; they do not divide values among copies.

Now transport the reward as well. If $v = W' + Dv$, set $v^+ = Pv$ and $W^+ = PW'$. Then

$$W^+ + C^+v^+ = PW' + C^+Pv = P(W' + Dv) = Pv = v^+.$$

The primary copy x_0 has $v^+(x_0) = v(x)$. An extra copy of an unflagged parent has value zero; this is compatible with its zero reward and zero outgoing row. We are not claiming that every formal copy preserves its parent's value.

Why the active count only doubles

Every active child has a parent in $\mathcal{A}(C)$: the support of D is contained in the original row and column supports, as follows from the correction recurrences. There is at most one primary active copy per such parent.

If a nonprimary child is active as a source, its row factor $r_i(x)$ is positive, so its parent is in O . If it is active as a destination, a positive split fraction reaches it, so its parent is again in O . Consequently

$$|\mathcal{A}(C^+)| \leq |\mathcal{A}(C)| + (m-1)|O| \leq 2|\mathcal{A}(C)|. \quad (38)$$

The full formal domain grows by m , but the active set grows by at most two. This distinction is indispensable: m was chosen larger than Kq , so a bound using m instead of two would not prove that active count times entry scale decreases.

Here is where the detector's witness count enters the algorithmic guarantee. Each flagged parent costs at most $m-1$ additional active vertices. The choice of R ensures that the total cost $(m-1)|O|$ is at most the previous active count. Meanwhile every entry shrinks by a factor of q . The product of the two bounds therefore improves by a factor of at least $q/2$. The corresponding smooth copying construction in the original manuscript is Proposition 4.5.

Recap

The new players are the copy number m , primary copies x_0 , activation factors r_i , routing fractions s_j , and the projection P . We proved entry reduction to f/q , active growth by at most two, and the value-preserving identity $C^+P = PD$. Only flagged parents can acquire extra active copies, and the detector's incidence count pays for them.

9 The complete hierarchy and its guarantee**Objective**

Assemble the pilot, correction, and copy operations into one algorithm. Track the target across changing vertex sets and prove that the accumulated reward approximates the original acceptance probability.

One stage, in executable order

The constants $q, \theta, k, \delta, m, R$ are fixed once as above. At a stage we are given a strictly forward substochastic matrix C with entries at most f , together with a reward vector W . The pilot calls the detector and produces the final gates; the actual correction produces E, D, W' on the old domain; copying produces C^+, W^+ on the new domain. The next box specifies the order, so there is no implicit oracle for an unknown target value.

Correction-and-copy stage with a direct-count detector.

1. **Pilot.** Starting at $\hat{E}_0 = 0$, perform the k updates (23)–(26). At each step, call the detector and accumulate its flags. Return the final flagged set O and gate g .
2. **Actual correction.** Freeze $g = \mathbf{1}_{V \setminus O}$, restart at $E_0 = 0$, and perform k updates (27). Set $E = E_k$.
3. **Expose reward.** Compute

$$D = C - E + EC, \quad W' = (I + E)W.$$

4. **Copy.** Form r_i, s_j, P , and C^+ using (32)–(36). Set

$$W^+ = PW', \quad f^+ = f/q.$$

Every operation is a finite computation on explicitly stored matrices and vectors. Neither the target vector nor a solution of a fixed-point equation is needed as input.

Keep the two iteration indices separate

The outer index ℓ labels a complete stage, whose input is C_ℓ, W_ℓ, f_ℓ . Inside that stage there are k pilot steps and k actual correction steps, indexed by i . When both indices are needed, write the actual corrections as $E_{\ell,i}$ and their final value as

$$E^{(\ell)} = E_{\ell,k}.$$

The scale $f_\ell = q^{-\ell}$ is constant throughout those inner steps. It is reduced only when we pass to the next outer stage. Similarly, the gate chosen at stage ℓ is fixed during that stage's actual correction, but a new pilot may choose different gates at stage $\ell + 1$.

Call structure and loops

Figure 12 separates the three levels of control. The *outer algorithm* invokes a fresh pilot at each stage. The *pilot loop* calls the detector at each of its k steps. The *actual correction loop* then runs for k steps without calling the detector. Only after both passes do we apply the correction and copy destinations.

The constant k is defined by (17); the total stage count L is chosen for the desired error in (41). For inverse-polynomial error and polynomial initial size, $L = O(\log n)$. Start with $\ell = 0$, $C_0 = S$, $W_0 = e$, and $f_0 = 1$. Within the two inner loops below, $C = C_\ell$ and $f = f_\ell$ are fixed; the pilot and actual pass each initialize their own index i to one.

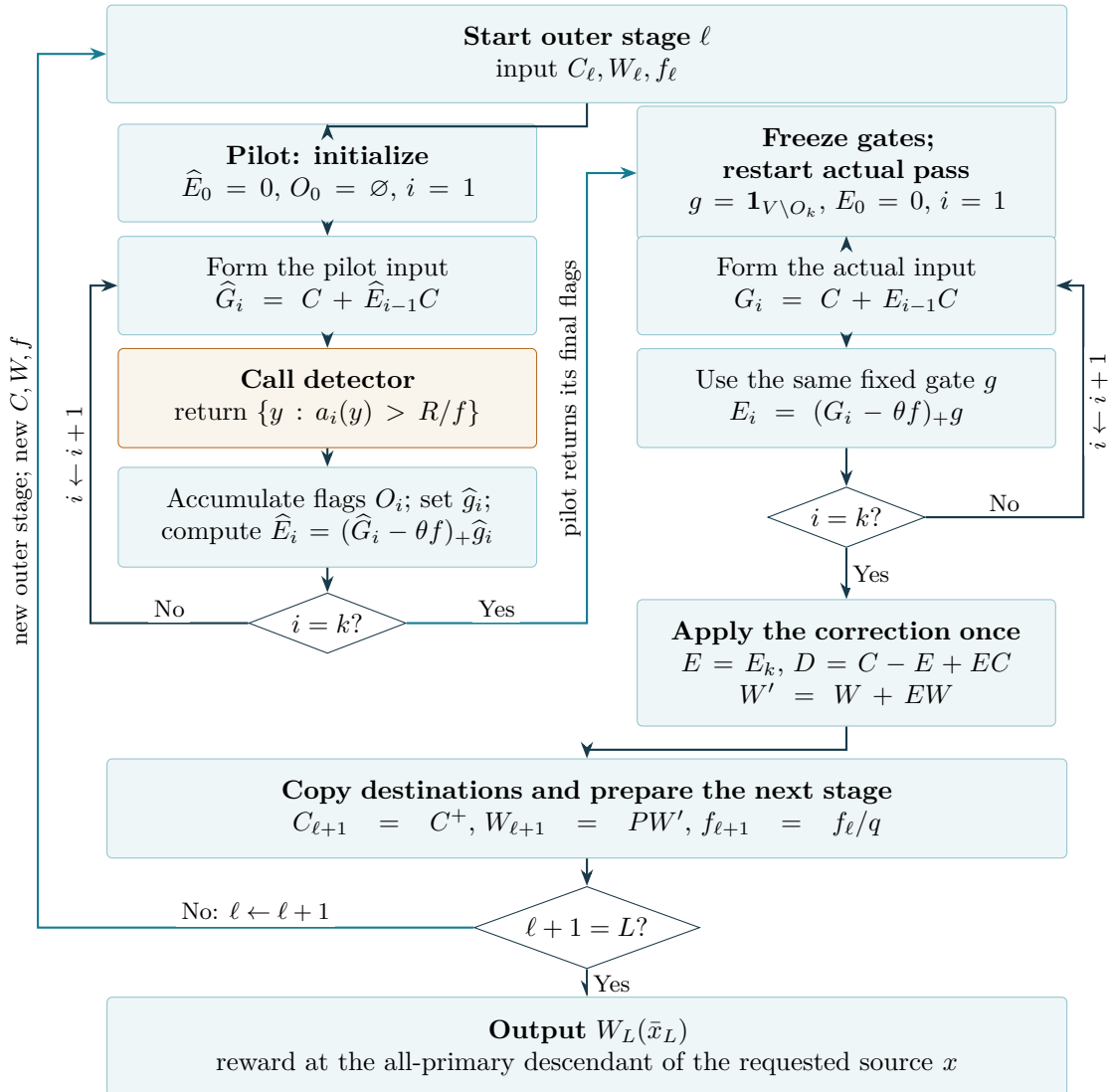


Figure 12: Who calls whom. Only the pilot calls the detector, highlighted in orange. The pilot remembers flags; the actual pass starts from zero with the final gates fixed. The inner loop index i is reset in each pass. The outer loop changes the matrix, reward, and scale, then starts a fresh pilot. Here $(G - \theta f)_+g$ denotes entrywise thresholding followed by columnwise gating.

The target changes its indexing, not its meaning

Initialize

$$V_0 = V, \quad C_0 = S, \quad W_0 = e, \quad f_0 = 1, \quad v_0 = p_0 = (I - S)^{-1}e.$$

After each stage, define

$$V_{\ell+1} = V_\ell \times \{0, \dots, m-1\}, \quad v_{\ell+1} = P_\ell v_\ell, \quad W_{\ell+1} = P_\ell(I + E^{(\ell)})W_\ell.$$

The *transported target* v_ℓ is a proof object used to track values on the enlarged vertex set; the algorithm need not compute it. The matrix P_ℓ is computed from that stage's flags and copy rule. We can therefore update $W_{\ell+1}$ from known quantities even though v_ℓ is unknown.

The correction identity and projection identity prove inductively that

$$\boxed{v_\ell = W_\ell + C_\ell v_\ell, \quad 0 \leq W_\ell \leq v_\ell \leq 1.} \quad (39)$$

Nonnegativity of W_ℓ follows from its update. The bound $v_\ell \leq 1$ follows because every row of P_ℓ has just one possible nonzero entry, of value at most one. The upper bound on W_ℓ follows from the identity and $C_\ell \geq 0$.

Let $\bar{x}_\ell = (x, 0, \dots, 0)$ be the *all-primary descendant* of a base vertex x . All projection factors on this chain are one, so

$$v_\ell(\bar{x}_\ell) = p_0(x).$$

This is the particular copy whose reward approximates the original value. For example, after two stages it is $((x, 0), 0)$, abbreviated $(x, 0, 0)$. Following primary copies never multiplies the target by zero. Other descendants can also have the same value if all their activation factors are one, but choosing the all-primary descendant gives a uniform rule that needs no such case distinction.

At these distinguished coordinates the reward is nondecreasing across stages:

$$W_{\ell+1}(\bar{x}_{\ell+1}) = W_\ell(\bar{x}_\ell) + (E^{(\ell)}W_\ell)(\bar{x}_\ell) \geq W_\ell(\bar{x}_\ell).$$

Thus we keep improving a lower approximation to the same original number $p_0(x)$, even though the surrounding matrix is indexed by a growing set.

The progress measure

Let $n_0 = |V_0|$. The stage bounds imply

$$|\mathcal{A}(C_\ell)| \leq 2^\ell |\mathcal{A}(S)| \leq 2^\ell n_0, \quad C_\ell(u, v) \leq q^{-\ell}.$$

There are two independent contributions to the error bound. Each remaining continuation entry is small, at most $q^{-\ell}$. The number of destinations that can contribute to a row is bounded by the active count, at most $2^\ell n_0$. Their product controls the row sum, and $0 \leq v_\ell \leq 1$ turns the row sum into a value-error bound. Applying (39) at the all-primary copy gives

$$\begin{aligned} 0 \leq p_0(x) - W_\ell(\bar{x}_\ell) &= (C_\ell v_\ell)(\bar{x}_\ell) \\ &\leq |\mathcal{A}(C_\ell)| q^{-\ell} \\ &\leq n_0 (2/q)^\ell. \end{aligned} \quad (40)$$

Stage	Entry scale	Active-count bound	Residual bound
0	1	n_0	n_0
1	q^{-1}	$2n_0$	$n_0(2/q)$
2	q^{-2}	$4n_0$	$n_0(2/q)^2$
ℓ	$q^{-\ell}$	$2^\ell n_0$	$n_0(2/q)^\ell$

These residual bounds can exceed one at early stages; the separate substochastic bound still gives an upper bound of one. Their purpose is to show eventual geometric decay. For $q = 16$, the displayed bound improves by a factor of eight per stage.

Theorem 9.1 (Exact dense hierarchy with column-controlled corrections). *Given a finite rational strictly forward substochastic matrix S and a nonnegative rational vector e with $p_0 = (I - S)^{-1}e \in [0, 1]^{V_0}$, the direct-count construction above produces, after every $\ell \geq 0$ stages, a nonnegative reward vector W_ℓ satisfying*

$$0 \leq p_0(x) - W_\ell(\bar{x}_\ell) \leq n_0(2/q)^\ell \quad (x \in V_0).$$

At stage ℓ , every actual correction column has at most R/f_ℓ nonzero entries, where $f_\ell = q^{-\ell}$. All stages are finite and deterministic in exact arithmetic.

For $0 < \varepsilon < 1$, it is enough to take

$$L = \left\lceil \frac{\log(n_0/\varepsilon)}{\log(q/2)} \right\rceil. \quad (41)$$

For inverse-polynomial error $\varepsilon = n^{-c}$ and an initial graph of size $n_0 \leq n^a$, where a, c are constants, this is $L = O(\log n)$. Explicitly, $L \geq (a + c) \log n / \log(q/2)$ suffices. The constants may be large, but they do not depend on the input length.

The full domain has size $|V_L| = n_0 m^L$, whereas the active-count bound is only $2^L n_0$. Both facts can be true at once: most formal copies may have no incident transitions. We still allow dense storage in this exposition. It is the mathematical active-count bound, not the allocated array size, that enters the approximation proof. This is the role played by Proposition 4.7 in the original manuscript.

In an acceptance-probability problem, the reward remains a one-sided approximation. If $p_0(x) = 0$, then every $W_\ell(\bar{x}_\ell)$ is exactly zero. If the original problem has a constant completeness-soundness gap, a sufficiently small additive error resolves that gap. A small-space implementation of this construction would be a separate theorem, and is outside these notes.

Recap

The full stage is pilot, fixed correction, reward update, then copying. The target on the current domain is v_ℓ ; its all-primary coordinates retain the original values $p_0(x)$. The central estimate is $n_0(2/q)^\ell$, giving inverse-polynomial accuracy after $\ell = O(\log n)$ stages when $n_0 = \text{poly}(n)$. We have proved a complete exact-arithmetic hierarchy with controlled correction columns, without sparse representations or statistical estimation.

10 Returning to the paper: what we simplified and why

Objective

Match the exposition to the supplied manuscript. Explain how the sampled detector fits the same argument, and distinguish mathematical support bounds from the sparse representations used by the later space-efficient implementation.

Which parts are the same?

The correction identity, the fixed-gate iteration, the ordered-support factorial argument, the pilot-and-restart structure, and the projection identity all follow the architecture of Section 4 of the manuscript. The progress estimate (40) is the same type of estimate as its Proposition 4.7.

We made three explicit simplifications:

1. We used the hard cutoff $Q(t) = (t - \theta)_+$ instead of the paper's smooth cutoff between $(t - \theta_2)_+$ and $(t - \theta_1)_+$.
2. We counted substantial column entries directly, instead of evaluating a detector built from random selected sets and order statistics.
3. We used binary activation and a hard routing threshold, instead of smooth activation and routing fractions.

These changes produce a variant with its own complete proofs above. They are not asserted to be drop-in replacements for the later small-space estimator. In particular, discontinuous counts and hard decisions cannot simply be substituted into an analysis that needs quantitative stability under perturbations.

The paper's detector: two sampled vertex sets, then an expectation

This subsection is a map back to the manuscript, not another prerequisite for the direct-count proof. Recall why sampling enters: the full implementation wants sparse sample-dependent objects it can access recursively with little memory. Its sampler is defined in the original Section 3; its pilot and detector are in Section 4; the associated matrix tables are built in Section 5.

At a fixed scale f , the paper uses two independent *channels* X and Y . A channel is a random seed together with a rule deciding which vertices are selected. It produces a whole subset, not one sampled vertex. Define

$$S_X = \{x : X_f(x) = 1\}, \quad S_Y = \{y : Y_f(y) = 1\}.$$

Each vertex is selected with probability π_f comparable to f . Independence here is between the channels; one should not assume that all vertex inclusion indicators within a channel are mutually independent.

There are two different questions to control. For a destination y , how many selected sources witness substantial incoming entries? For a source x , how many selected destinations make its row crowded? Channel X serves the first question, and channel Y the second. This is the context for the two sets in the formula below.

Let G_i denote the paper's deterministic pilot input at step i ; it corresponds to our $\hat{G}_i$. The terminal channel Y first defines a *row gate*

$$R_i(x) = \rho(\text{rank}_{K_{\text{row}}} \{G_i(x, w)/f : Y_f(w) = 1\}). \quad (42)$$

Here $\text{rank}_{K_{\text{row}}}$ means the K_{row} th largest number, with zeros added if fewer numbers are present. The manuscript calls this rank parameter K ; we write K_{row} to avoid confusing it with our rough entry-bound constant $K = k + 1$.

An *order statistic* is a smooth-compatible way to ask for many witnesses. For example, the third largest of 0.9, 0.8, 0.2, 0.1 is 0.2: at least three entries reach 0.2, but fewer than three reach 0.5. Thus a large K_{row} th entry means that many selected destinations have substantial entries in this row. The decreasing cutoff ρ is one when that statistic is small and zero when it is too large. The resulting row gate $R_i(x) \in [0, 1]$ suppresses a row that is too crowded in this trial.

The row gate $R_i(x)$ has a different role from the column gate $g(y)$. It controls whether source x contributes to one sampled detector trial. The column gate $g(y)$ controls whether the actual correction is enabled at destination y . The direct-count algorithm needed only the latter because it imposed no requirement on the representation of an individual trial row.

Next, a rising cutoff ϕ turns an entry $G_i(x, y)/f$ into a score indicating whether it is substantial. The source channel X selects candidate witnesses x for a column y . The deterministic detector value is

$$U_i(y) = \mathbb{E}[\text{rank}_J\{\phi(G_i(x, y)/f)R_i(x) : X_f(x) = 1\} \mid Y_f(y) = 1]. \quad (43)$$

Read this expression in four steps. Fix the destination y and condition on its being in the selected terminal set. For each selected source $x \in S_X$, form its substantial-entry score $\phi(G_i(x, y)/f)$ and multiply it by its row gate $R_i(x)$. Take the J th largest of these scores. Finally average that statistic over the channel randomness under the stated conditioning.

The rank J asks for many substantial source witnesses that survive their row gates. If there are fewer than J selected sources, zero padding makes the statistic zero. If the scores were all binary, a statistic of one would mean that at least J witnesses survived; the smooth scores interpolate this test.

The final number $U_i(y)$ is a *detector score*, not an estimate of the acceptance probability $p(y)$. It is a true conditional expectation and therefore deterministic, even though its definition uses random subsets. With no computational restrictions, it can be evaluated by enumerating the finite channel space with its prescribed probabilities. The later small-space implementation estimates such quantities; that is a separate layer of the manuscript.

The two channels have different jobs: X samples the incoming witnesses to a column, while Y keeps the trial's row representation under control. In our direct-count construction, no trial row needs to be represented by a short list, so we removed both channels and counted the column witnesses in the full matrix.

The key guarantees proved for the sampled detector have the same two directions that our direct count provided:

- sufficiently many large entries in a column force a high detector score, so that column is disabled for correction;
- a noticeably positive detector score certifies many substantial entries, so the number of columns that can activate extra copies is small.

The second direction is easy to overlook. A detector that flags every column would make all correction supports zero, but would require far too many copies. The detector must both protect the incoming-support bound and justify the cost of the columns it disables. These are the sampled counterparts of the two proofs in Section 7: domination bounds correction supports, while witness counting bounds the flagged set.

Why the paper smooths the gates and copying rule

The manuscript uses a *running maximum* $U^{\max}(y) = \max_i U_i(y)$ and a decreasing cutoff γ to set $g(y) = \gamma(U^{\max}(y))$. It also uses a rising cutoff κ for nonprimary activation factors and smooth routing fractions depending on $D(x, y)/f$.

The threshold intervals are arranged so that a nonprimary fraction can be positive only when its destination's activation factor is already exactly one. This is precisely the condition (35) that made $C^+P = PD$ work. Smooth values of $r_i(x)$ between zero and one are allowed; then the corresponding copy has value $r_i(x)v(x)$. The projection P is genuinely weighted in that version.

Why add this smoothness when a hard decision is easier to describe? In our exact algorithm we know on which side of every threshold an entry lies. An estimator may return a nearby value instead. A discontinuous flag can change completely under an arbitrarily small perturbation near its threshold. The smooth version allows quantitative control of how such input errors affect later objects, while separated threshold intervals preserve the exact routing-and-activation implication.

The smooth cutoffs have an additional property: their eighth roots, and the relevant complementary roots, are Lipschitz on the needed ranges. This property supports the manuscript's later eighth-moment error analysis. It is not required for our exact hard-threshold proofs, but explains why our simpler rules should not be substituted unchanged into the full estimator.

Where the sampler parameters went

For the exact hierarchy, the shrinkage parameter is simply q . In the manuscript, $q = 2^H$ and the stage scale is

$$f_\ell = q^{-\ell} = 2^{-H\ell} = b_\ell.$$

The index ℓ is again the outer stage number. The manuscript defines a whole *grid of sampling rates*, $b_j = 2^{-Hj}$. Its stage- ℓ scale is the particular grid point b_ℓ . Other parts of the implementation may use other grid points. We write the generic grid index as j here so it is not confused with our fixed inner iteration count k .

The integer H is fixed once, independently of the input size. It both determines the geometric ratio $q = 2^H$ and serves as the *dimension* in the algebraic sampler. The auxiliary constant h_0 , with $2 \leq h_0 < H$, belongs to that sampler's algebraic construction. The exact dense hierarchy only needs the ratio q ; it never manipulates H -dimensional samples or uses h_0 .

The paper also uses a *global label bound* $N = 2^{C_N B}$, where B is its size parameter for logarithmic-space accounting. Here C_N is a fixed constant chosen to make the identifier universe large enough; it is not a quantity depending on N . Its subscript names its role in the bound for N . Neither N nor C_N is an entry scale. In our exposition the actual initial size is n_0 , the full formal size at stage ℓ is $n_0 m^\ell$, and the active-count bound is $2^\ell n_0$. Keeping these three counts separate avoids confusing address size with remaining continuation mass.

What the later sections of the paper add

Our matrices can be large, and their entries are themselves recursively defined. Return to the example $(EC)(x, y) = \sum_z E(x, z)C(z, y)$ from Section 5: having an exact recurrence does not yet tell us how to navigate its nested computations with a small memory footprint. The later algorithm must generate local information when needed, control approximation errors, and recover enough context to return from recursive work.

Here is a compact map of that implementation. The sparse sample-dependent tables and their normalization are in original Section 5. Conditional averaging and recursive estimation are

in Sections 6–7. Sections 8–11 handle compression, fingerprints, numerical precision, and the controller that executes the computation within its memory budget. These are descriptions of the roles of those sections, not additional steps hidden inside our dense algorithm.

We have omitted all of that implementation. We have *not* omitted support bounds as mathematical facts: the row bound (14), incoming bound (22), and active bound (38) are part of why the exact hierarchy works. A support bound can be useful in a proof even when the algorithm stores a dense matrix.

The permutation-BP restriction is not needed for any proof in these notes. A permutation BP gives an especially simple starting transition system, but the correction mechanism applies to general ordered substochastic matrices with a bounded target vector.

A map of the players

Object	Role in these notes	Where it appears
C, W, p	Continuation, exposed reward, and target value; $p = W + Cp$.	Sections 1–2
E, D	Correction and its residual $D = C - E + EC$.	Sections 2–4
f, θ, q	Current entry scale, reserve threshold, and stage shrinkage factor.	Sections 3–5
E^*, T, A_x, Δ_k	Proof objects that produce the factorial truncation bound.	Section 4
g	Gate fixed during the actual correction; chooses enabled columns.	Sections 3–7
$\hat{E}_i, \hat{G}_i$	Pilot corrections and inputs used to choose g .	Section 7
a_i, O_i, O	Direct column counts and the accumulated overload flags.	Section 7
m, r_i, s_j, P	Copies, source activation, destination routing, and value transport.	Section 8
$v_\ell, \bar{x}_\ell$	Target on the current domain and the all-primary copy preserving the base value.	Section 9
X, Y, R_i, U_i	Selected vertex sets, row gates, and detector expectations in the manuscript’s version.	Section 10

Source guide

In the supplied manuscript, the main references are Lemma 2.1 (transition model), Lemma 4.1 (factorial truncation), Lemmas 4.2–4.3 (pilot and detector), Lemma 4.4 (finite residual), Proposition 4.5 (copying), and Proposition 4.7 (reward transport). The direct-count detector and hard routing are pedagogical variants proved in these notes; the statistical implementation in Sections 5–13 is outside their scope.

Recap

The exact mechanism has two layers. First, thresholded correction transfers value into reward and gives entry decay. Second, the pilot uses the detector to restrict incoming correction support, and copying repairs the columns the pilot disables. The sampled detector, smooth gates, and sparse representations support the later effort to evaluate that mechanism with little memory. The central invariants to carry back to the paper are $p = W + Cp$, $I - D = (I + E)(I - C)$, $C^+P = PD$, and the decrease of active count times entry scale.