Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Algebraic Expressions for Directed Grid Graphs with Diagonal Edges: Decomposition Bounds, Lower Bounds, and Algebraic-Branching-Program Methods

Mark Korenblit

korenblit@hit.ac.il

Holon Institute of Technology

Vadim E. Levit

Ariel University

Research Article

Keywords: st-dag expressions, algebraic branching programs, triangulated grid graphs, king graphs, fractal grid graphs, min-cuts, network reliability, decomposition algorithms, path polynomials, formula lower bounds

Posted Date: July 29th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9728120/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Algebraic Expressions for Directed Grid Graphs with Diagonal Edges: Decomposition Bounds, Lower Bounds, and Algebraic-Branching-Program Methods

Mark Korenblit[1*] and Vadim E. Levit[2]

[1*]School of Computer Sciences, Holon Institute of Technology, Golomb, Holon, 5810201, Israel.
[2]Department of Mathematics, Ariel University, Kiryat Hamada, Ariel, 40700, Israel.

*Corresponding author(s). E-mail(s): korenblit@hit.ac.il;
Contributing authors: levitv@ariel.ac.il;

Abstract

We study formal path expressions for two edge-labeled two-terminal directed acyclic graph families: directed triangulated grid graphs (TGGs) and directed king graphs. The expression is the path polynomial, namely the formal sum of all source-to-target path products in the free noncommutative semiring, and its length is the number of label occurrences in an explicit formula. For TGGs with fixed depth m and variable size n, the backtracking construction has length $O_m(n^m)$, while decomposition and alternating decomposition give $O_m(n \log^{m-1} n)$. We prove that this bound is globally optimal for depths $m \leq 4$ by a binomial-language projection. For every fixed depth, we also prove a matching lower bound within a recursively defined balanced interval-decomposition model that contains the recursive algorithms. For directed king graphs, backtracking is exponential already in depth 2. Geometric decomposition gives $O_m(n^{\log_2(4m-2)})$, and a column-transfer interpretation as a fixed-width algebraic branching program (ABP) improves this to $O_m(n^{1+\log_2 m})$. Parity-language restrictions give an unrestricted $\Omega(n^2)$ lower bound, tight for depth 2, and iterated-matrix-multiplication restrictions give product-depth lower bounds. The final part relates path-polynomial factorizations to min-cuts and two-terminal reliability through zero substitutions.

Keywords: st-dag expressions, algebraic branching programs, triangulated grid graphs, king graphs, fractal grid graphs, min-cuts, network reliability, decomposition algorithms, path polynomials, formula lower bounds

MSC Classification: Primary 68R10; Secondary 05C85 , 68Q25 , 68Q45 , 68Q17 , 68W30

1 Introduction

A *graph* G consists of a *vertex set* $V(G)$ and an *edge set* $E(G)$, where each edge corresponds to a pair (v, w) of vertices. If the edges are ordered pairs of vertices (i.e., the pair (v, w) is different from the pair (w, v)), then we call the graph *directed* or *digraph*; otherwise, we call it *undirected*. If (v, w) is an edge in a digraph, we say that (v, w) *leaves* vertex v and *enters* vertex w. A vertex in a digraph is a *source* if no edges enter it, and a *target* if no edges leave it.

A *path* from vertex v_0 to vertex v_k in a graph G is a sequence of its vertices $[v_0, v_1, v_2, \ldots, v_{k-1}, v_k]$ such that $(v_{i-1}, v_i) \in E(G)$ for $1 \leq i \leq k$. G is an *acyclic graph* if there is no closed path $[v_0, v_1, v_2, \ldots, v_k, v_0]$ in G. A *two-terminal directed acyclic graph* (*st-dag*) has only one source s and only one target t. In an st-dag, every vertex lies on some path from s to t (an *s-t-path*).

For an st-dag G, a set $C \subseteq E(G)$ is an *s-t cut* if every s-t path contains at least one edge of C, or equivalently, if deleting C destroys all directed paths from s to t. An s-t cut C is *inclusion-wise minimal* if no proper subset of C is an s-t cut. In the reliability literature such inclusion-wise minimal cuts are often called *min-cuts*. This terminology is used below in the cut-substitution theorem and in the discussion of reliability and fractal grid graphs.

A graph G' is a *subgraph* of G if $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. A graph G' is *homeomorphic* to a graph G (a *homeomorph* of G) if G' can be obtained by subdividing edges of G with new vertices (replacing corresponding edges with paths of new edges). Two graphs G and G' are *isomorphic* if there exists a bijection $f : V(G) \to V(G')$ such that $(v, w) \in E(G)$ if and only if $(f(v), f(w)) \in E(G')$.

Given a graph G, let $X_G = \{x_e \mid e \in E(G)\}$ be a set of formal edge labels, one label for each edge. Throughout the paper, expressions are interpreted in the free non-commutative semiring $\mathbb{N}\langle X_G \rangle$. Addition is formal sum, multiplication is concatenation of words, 1 is the empty word, and no cancellation or idempotent law is used. Thus all length estimates below count occurrences of formal labels and are not affected by accidental identities that may hold in a particular ring. In what follows, elements of X_G are called *labels*, and a *labeled graph* refers to an edge-labeled graph with all labels distinct.

Two algebraic expressions F_1 and F_2 are *algebraically equivalent* if they define the same element of $\mathbb{N}\langle X_G \rangle$; equivalently, one can be transformed into the other using associativity of $+$ and $\cdot$, commutativity of $+$, distributivity, and the identities for 0 and 1. We regard F_1 as a *representation* of F_2 and F_2 as a *representation* of F_1. For instance, $a(b + c) + ad$ is a representation of $a(b + d) + ac$ and vice versa.

Each s-t-path in an st-dag can be represented as the product of all edge labels along the path. The label order in every product (from the left to the right) is identical to the order of corresponding edges in the path (from the source to the target). We define the sum of edge label products corresponding to all possible s-t-paths of an st-dag G as the *canonical expression* of G. This is the same path-sum polynomial that is computed by an algebraic branching program (ABP), except that here the given st-dag itself is fixed and the edge labels are formal noncommuting variables [4, 15, 49]. An algebraic expression is called an *st-dag expression* (a *factoring of an st-dag* in [3]) if it is algebraically equivalent to the canonical expression of an st-dag. An st-dag expression consists of labels, the operations + and ·, and parentheses. We denote an expression of an st-dag G by $Ex(G)$.

For example, the canonical expression of the st-dag presented in Figure 1 is $abd + abe + acd + ace + fe + fd$. All its representations (e.g., $a(b + c)(d + e) + f(d + e)$, $(ab + ac + f)(d + e)$ and others) are st-dag expressions. On the other hand, $ab + bc$ is not an st-dag expression of any graph.

We define the number of label occurrences in an st-dag expression as its *length*. For example, the lengths of the above expressions for the st-dag in Figure 1 are 16, 8, 7, respectively. An *optimal representation of the expression* F is an expression of minimum length algebraically equivalent to F. In this paper we construct compact representations produced by specified recursive algorithms and analyze their lengths. Except where explicitly stated, the bounds below are bounds for the expressions generated by these algorithms, not claims of global optimality.

Remark 1 (Formula length versus shared syntax DAGs) All lower and upper bounds in the paper are stated for formulas or explicit expression strings, i.e., for tree representations in which repeated subexpressions are counted each time they occur. An implementation may store repeated subexpressions as a shared syntax DAG, but then the storage measure is different from the formula length analyzed here. Thus the formula-length theorems should not be read as bounds on the size of a maximally shared circuit representation.

A *series-parallel graph* is defined recursively as follows:

(i) A single edge (u, v) is a series-parallel graph with source u and target v.

(ii) If G_1 and G_2 are series-parallel graphs, so is the graph obtained by either of the following operations:

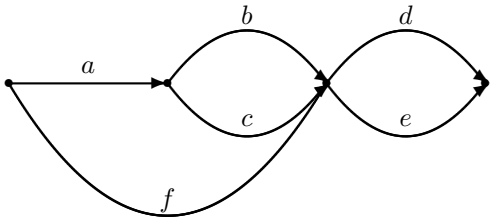
(a) *Parallel composition*: identify the source of G_1 with the source of G_2 and the target of G_1 with the target of G_2.

(b) *Series composition*: identify the target of G_1 with the source of G_2.

A series-parallel graph expression has a representation in which each label appears only once [3, 31] (a *read-once formula* [16] in which Boolean operations are replaced by their arithmetic counterparts). This representation is an optimal representation of the series-parallel graph expression. For example, an expression of the series-parallel graph presented in Figure 1 can be reduced to the optimal representation $(a(b+c)+f)(d+e)$.

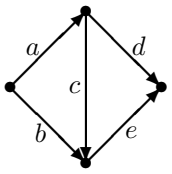
Using the results obtained in [10] for undirected graphs, Burns and Steiner [7] showed that an st-dag is series-parallel if and only if it does not contain a subgraph homeomorphic to the *forbidden graph* (*Wheatstone bridge*) illustrated in Figure 2.

Fig. 1 An example of a series-parallel graph.

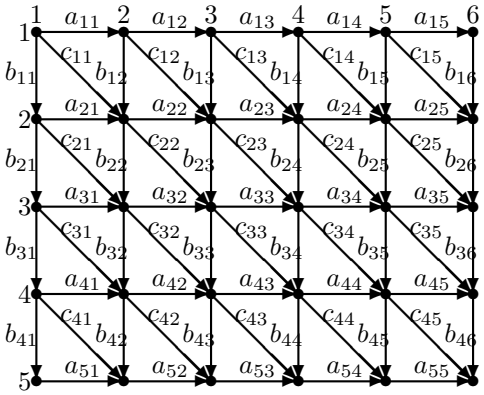


Possible optimal representations of its expression are $a\,(ce+d)+be$ or $(ac+b)\,e+ad$. That is, this graph has no expression in which each label appears exactly once. For this reason, an expression of a non-series-parallel st-dag cannot be represented as a read-once formula. However, generating an optimum factored form for an arbitrary expression is NP-hard [45].

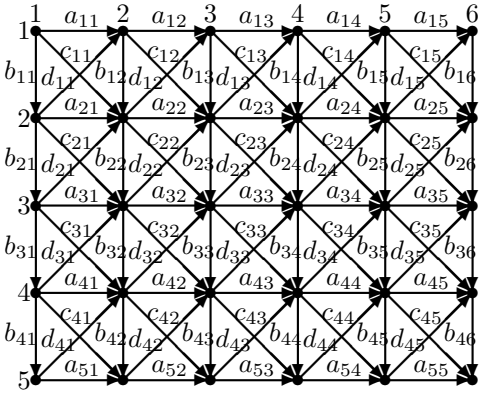
Fig. 2 The forbidden subgraph for series-parallel graphs.



Computations on graphs whose edges carry additional data have applications in many areas. In particular, network problems arising in scheduling [7, 14, 57], reliability [51, 53, 56], and economics or process analysis [50, 61] are often difficult in general but admit efficient solutions on series-parallel graphs. Optimization problems on vertex-labeled graphs are studied in [13, 21, 52].



(a) An example of a triangulated grid graph



(b) An example of a king graph

Fig. 3 Directed grid graphs with diagonal edges. Horizontal, vertical, down-right diagonal and up-right diagonal edges are labeled by a, b, c and d, respectively.

In [28] a directed *grid graph* $G_{m,n}$ [58] was studied. It has $m \times n$ vertices and $m(n-1)+n(m-1)$ edges. Each vertex corresponds to a pair of integer coordinates (i, j) with $1 \leq i \leq m$ and $1 \leq j \leq n$. Its edge set is $\{((i,j),(i+1,j)) \mid 1 \leq i < m, 1 \leq j \leq n\} \cup \{((i,j),(i,j+1)) \mid 1 \leq i \leq m, 1 \leq j < n\}$. Each horizontal edge $((i,j),(i,j+1))$ is labeled a_{ij}, and each vertical edge $((i,j),(i+1,j))$ is labeled b_{ij}. Throughout this paper, m is treated as a fixed *depth* parameter, while n is the *size* parameter.

The present paper focuses on grid graphs augmented with diagonal edges. Such graphs arise naturally in the study of maze-solving and reachability problems [23]. The first family is the directed *triangulated grid graph* (*TGG*) $T_{m,n}$ [41, 59]. It has the same vertices and edges as $G_{m,n}$ and additionally includes edges $\{((i,j),(i+1,j+1)) \mid 1 \leq i < m, 1 \leq j < n\}$ labeled c_{ij}. The second family is the directed *king graph* $K_{m,n}$ [43, 60]; in addition to all vertices and edges of $T_{m,n}$, it also includes edges $\{((i,j),(i-1,j+1)) \mid 2 \leq i \leq m, 1 \leq j < n\}$ labeled $d_{i-1,j}$. Examples of $T_{5,6}$ and $K_{5,6}$ are shown in Figure 3(a) and Figure 3(b), respectively.

The reliability connection is especially relevant for two-terminal directed acyclic networks. In a probabilistic st-dag, each edge may be operational or failed, and two-terminal reliability asks for the probability that at least one operational source-target path remains. Inclusion-wise minimal cuts are then minimal failure certificates. Their enumeration is a classical component of network-reliability algorithms, and Provan and Ball showed that reliability can be computed in time polynomial in the number of source-target cuts [51]. This places the growth of min-cut families next to the expression length questions studied here: both measure, in different ways, how complicated the same source-target path structure is.

Interrelations between graphs and expressions are discussed in [2, 3, 5, 9, 12, 17, 20, 45–48, 50, 54] and related work. Many of these papers develop algorithms for obtaining good factored forms of graph-related expressions; this is especially important in logic synthesis [17, 45].

Several previous works studied algebraic expressions for special non-series-parallel st-dags. For example, [31] gives an algorithm that generates an expression of length $O\left(n^2\right)$ for an n-vertex Fibonacci graph [18], a standard non-series-parallel example. More complicated rhomboidal graphs were considered in [27]; for these n-vertex graphs, the derived expression lengths are $O\left(n^{\log_2 6}\right)$.

There is also a related reliability and min-cut line in the earlier work of Korenblit and Levit. For Fibonacci graphs they studied st-connectedness and the number and sizes of min-cuts [30, 32], and they computed reliability polynomials for Fibonacci and related graphs [33]. For general two-terminal directed acyclic graphs they investigated minimal cuts and nested graphs [34, 35], while generalized Fibonacci graphs of degree 3 were treated in [36]. This earlier line is not a separate application appended to the present paper; it explains why the path-expression viewpoint and the cut-enumeration viewpoint belong to the same source-target algebra.

A recent continuation of this cut-enumeration viewpoint was presented for *fractal grid graphs* in [40]. In that work, a fractal grid graph is understood as a directed graph with recursive or self-similar grid-like structure, and the families considered include generalized Fibonacci graphs, directed grid graphs, and directed trapezoidal

grid graphs. The common parameter is the *range*: the reported number of inclusion-wise minimal source-target cuts grows linearly with the number of vertices for range 1, quadratically for range 2, and exponentially for range greater than 2. This is closely related to the present paper because the canonical path polynomial records all source-target paths. As shown in Theorem 4 below, a set of edges is an inclusion-wise minimal source-target cut precisely when setting its labels to 0 is an inclusion-wise minimal way to annihilate the path polynomial. Thus path factorization and minimal-cut enumeration give two complementary algebraic views of the same source-target structure: one compresses the set of all paths, while the other enumerates minimal blockers of all paths. In reliability language, these blockers are precisely minimal failure configurations.

Our goal is to construct compact expressions for directed TGGs and directed king graphs and to prove bounds on the lengths of the expressions produced by the specified algorithms. We use two recursive methods: the *backtracking method* and the *decomposition method*. Most upper bounds below are algorithm-specific output bounds. However, Section 2 also proves general lower bounds. In particular, the TGG decomposition bound is shown to be asymptotically optimal globally for depths up to four and optimal for all fixed depths inside a precisely defined balanced interval-decomposition model. The depth-two king-graph formula bound is shown to be tight. General monotonicity questions for shortest st-dag expressions of subgraphs were studied in [39].

Notation for method-specific lengths. The symbol $L_m(n)$ is used locally in each algorithmic section for the length produced by the method under discussion. When several methods are compared, the following optional superscripts clarify the meaning:

$$L_m^{\mathrm{bt}}(n), \quad L_m^{\mathrm{dec}}(n), \quad L_m^{\mathrm{imp}}(n), \quad L_m^{\mathrm{alt}}(n),$$

for backtracking, basic decomposition, improved decomposition, and alternating decomposition, respectively. The local notation $L_m(n)$ is kept in the theorem statements to avoid overburdening formulas.

Relation to the conference version. An earlier conference version appeared in [29]. The expanded title of the present article is intended to signal the added lower-bound and algebraic-branching-program components. The present article is a substantially revised journal version. The conference version introduced the two diagonal grid families and the basic recursive constructions. The current version makes the formal algebraic model explicit by working in the free noncommutative semiring; supplies full recurrence proofs, including a canonical-transposition and naturality proof of the symmetry $L_m(n) = L_n(m)$ for backtracking; corrects boundary and indexing cases in the king graph definition and in the small TGG examples; states the alternating decomposition rule precisely and separates its symmetry from graph isomorphism alone; replaces the sketch of the king-graph decomposition bound by a uniform proof over rectangular, upper-cut, under-cut, and two-cut subgraphs; adds lower bounds based on regular-expression complexity of binomial and parity languages, exact small base cases, an all-depth balanced interval-decomposition optimality theorem for TGGs, a column-transfer algebraic-branching-program simulation for king graphs, an unrestricted quadratic lower bound for king graphs, and a depth-restricted lower bound via

iterated matrix multiplication; and expands the literature positioning, bibliography, and applications discussion.

Although a graph under consideration may be a subgraph of another graph, we assume that, when representing expressions of TGGs and king graphs, these graphs are initial and their vertices and edges are numbered starting from 1, as in Figure 3.

2 General Lower Bounds and Restrictions

Before analyzing particular construction algorithms, we record several lower bounds that hold for all expressions in the formal semiring model. Let P_G denote the canonical path polynomial of an edge-labeled st-dag G, and let $\ell(P_G)$ be the minimum possible number of label occurrences among all expressions equivalent to P_G in $\mathbb{N}\langle X_G\rangle$. Thus the algorithmic quantities denoted by $L_m(n)$ in later sections are upper bounds on $\ell(P_G)$ for the corresponding graph family, and equality is asserted only in the special cases explicitly proved below.

Theorem 1 (edge-occurrence lower bound) *Let G be an edge-labeled st-dag in which every edge lies on at least one s-t path, and suppose all edge labels are distinct. Then*

$$\ell(P_G) \geq |E(G)|.$$

Proof Let x_e be the label of an edge e. Since e lies on an s-t path, there is at least one monomial of P_G containing x_e. If an expression contains no occurrence of x_e, then every monomial obtained after expanding that expression belongs to the free subsemiring generated by $X_G \setminus \{x_e\}$ and therefore contains no x_e. Hence every edge label must occur at least once in any expression for P_G. Since the labels are distinct, at least $|E(G)|$ label occurrences are necessary. □

Proposition 2 *For triangulated grid graphs and directed king graphs,*

$$\ell(P_{T_{m,n}}) \geq 3mn - 2m - 2n + 1$$

and, with the edge set of $K_{m,n}$ defined as in Section 1,

$$\ell(P_{K_{m,n}}) \geq 4mn - 3m - 3n + 2.$$

Proof In $T_{m,n}$ there are $m(n-1)$ horizontal edges, $(m-1)n$ vertical edges, and $(m-1)(n-1)$ downward diagonal edges. Their sum is $3mn - 2m - 2n + 1$. In $K_{m,n}$ there are additionally $(m-1)(n-1)$ upward diagonal edges, giving $4mn-3m-3n+2$. Every such edge is contained in an s-t path, so the result follows from Theorem 1. □

The next observation is the formal replacement for informal subgraph monotonicity. It will be used later to transfer lower bounds from hard restricted subgraphs.

Theorem 3 (restriction monotonicity) *Let G be an edge-labeled st-dag. Suppose a semiring homomorphism σ obtained by assigning some labels to 0, some labels to 1, and possibly renaming the remaining labels maps P_G to a polynomial P_H. Then*

$$\ell(P_G) \geq \ell(P_H).$$

In particular, lower bounds for a polynomial obtained from P_G by deleting edges and contracting forced labeled paths are also lower bounds for P_G.

Proof Let F be an expression of minimum length for P_G. Applying σ to all labels in F gives an expression for P_H. Deleting occurrences of the constants 0 and 1, and simplifying the corresponding zero summands and unit factors, cannot increase the number of label occurrences. Therefore P_H has an expression of length at most $\ell(P_G)$. □

The same substitution language gives a concise algebraic interpretation of source-target cuts. This will be used as a structural and reliability-theoretic bridge to the fractal-grid min-cut work cited in the introduction, but it is worth making explicit.

Theorem 4 (cut substitutions) *Let G be an edge-labeled st-dag and let $C \subseteq E(G)$. Let σ_C be the homomorphism of $\mathbb{N}\langle X_G \rangle$ defined by*

$$\sigma_C(x_e) = 0 \quad (e \in C), \qquad \sigma_C(x_e) = x_e \quad (e \notin C).$$

Then $\sigma_C(P_G) = 0$ if and only if C intersects every s-t path of G. Consequently, C is an inclusion-wise minimal s-t cut if and only if $\sigma_C(P_G) = 0$ and $\sigma_D(P_G) \neq 0$ for every proper subset $D \subset C$.

Proof The polynomial P_G is the formal sum of the monomials corresponding to all s-t paths of G. After applying σ_C, the monomial of a path Q vanishes exactly when Q contains at least one edge of C. Since the semiring has no cancellation, the whole polynomial becomes 0 exactly when every source-target path contains an edge of C, i.e., exactly when removing C disconnects t from s. The statement about inclusion-wise minimality is the same equivalence applied to all proper subsets of C. □

Theorem 4 explains why the min-cut growth phenomena for fractal grid graphs in [40] are naturally adjacent to the formula-length questions studied here. The present paper does not enumerate minimal cuts of $T_{m,n}$ or $K_{m,n}$; instead it studies the size of factored expressions for their path polynomials. Nevertheless, both problems are governed by the same recursive source-target geometry. In particular, the range transition reported in [40]–linear, quadratic, and then exponential growth of the number of inclusion-wise minimal cuts–suggests that cut-complexity and expression-complexity should be compared systematically for broader self-similar grid families.

This comparison should be interpreted carefully. The path polynomial is not, by itself, the reliability polynomial: path events overlap, and reliability is a probability of a union of operating-path events. The relevance is more basic and more robust. The formal path polynomial is a support object for all source-target paths; its minimal zero-substitution sets are exactly the minimal edge-failure sets disconnecting the target from the source. Thus the same factorized expression can be used on the path

side, while its annihilating sets describe the minimal cut side that underlies cut-based reliability methods.

For a finite language L, let $\mathrm{aw}(L)$ denote the minimum number of alphabet-symbol occurrences in a regular expression for L, using only union and concatenation. For $N \geq 1$ and $0 \leq k \leq N$, put

$$\mathcal{B}_{N,k} = \{w \in \{0,1\}^N : |w|_1 = k\}.$$

These are the standard binomial languages. We use the following consequence of Mousavi's linear-program lower-bound analysis for binomial languages: in Section 7 of [44], a feasible solution of the relaxed program $P'(N,k)$ with objective value $\Omega(N \log^k N)$ is constructed for $k \leq 3$, and hence the Ellul–Krawetz–Shallit–Wang expressions for $\mathcal{B}_{N,k}$ are asymptotically optimal in these cases. Equivalently, for each fixed $k \leq 3$,

$$\mathrm{aw}(\mathcal{B}_{N,k}) = \Omega_k(N \log^k N).$$

This is the exact range needed below, because the depth parameter in $T_{m,n}$ gives $k = m - 1$. The later work of Cseresnyes and Seiwert places this and related lower bounds in a regular-expression-to-arithmetic-formula framework, and also separates the all-fixed-k strengthening from the cases used here [6, Sec. 5.1].

Theorem 5 (global TGG lower bound for depths at most four) *For $m \in \{1,2,3,4\}$,*

$$\ell(P_{T_{m,n}}) = \Omega_m\left(n \log^{m-1} n\right).$$

Proof For $m = 1$ the claim is $\ell(P_{T_{1,n}}) = \Omega(n)$, which follows from Theorem 1. Assume $2 \leq m \leq 4$ and let F be an arbitrary expression for $P_{T_{m,n}}$. Define a homomorphism from edge labels to binary words by

$$a_{i,j} \mapsto 0, \qquad b_{i,j} \mapsto 1, \qquad c_{i,j} \mapsto 10.$$

After applying this homomorphism to F, and reading $+$ as union and multiplication as concatenation, we obtain a regular expression over $\{0,1\}$. Each source-target path in $T_{m,n}$ advances exactly $n-1$ columns and $m-1$ rows; a diagonal edge contributes one column advance and one row advance. Therefore every image word has length $N = n + m - 2$ and exactly $k = m - 1$ ones. Conversely, every word in $\mathcal{B}_{N,k}$ is realized by the path that uses only horizontal a-edges for the zeros and vertical b-edges for the ones. Hence the image language is precisely $\mathcal{B}_{N,m-1}$.

Each occurrence of an a- or b-label contributes one alphabet symbol to the regular expression, and each occurrence of a c-label contributes two. Thus

$$\mathrm{aw}(\mathcal{B}_{N,m-1}) \leq 2|F|.$$

Since $m - 1 \leq 3$, Mousavi's lower bound for binomial languages gives

$$|F| \geq \frac{1}{2}\,\mathrm{aw}(\mathcal{B}_{N,m-1}) = \Omega_m(N \log^{m-1} N) = \Omega_m(n \log^{m-1} n).$$

Taking the minimum over all expressions F proves the theorem. □

Conjecture 1 (binomial-language route to all fixed TGG depths) For every fixed $k \geq 0$,

$$\mathrm{aw}(\mathcal{B}_{N,k}) = \Omega_k(N \log^k N).$$

If this conjecture holds, then the same projection proof gives

$$\ell(P_{T_{m,n}}) = \Omega_m\left(n \log^{m-1} n\right)$$

for every fixed m.

Proposition 6 (exact small cases) *The following optimal lengths hold:*

$$\ell(P_{T_{m,1}}) = m - 1, \qquad \ell(P_{T_{1,n}}) = n - 1, \qquad \ell(P_{T_{2,2}}) = 5,$$

and

$$\ell(P_{K_{2,2}}) = 7.$$

Proof The first two polynomials consist of one path product of length $m-1$ or $n-1$, respectively. For $T_{2,2}$,

$$P_{T_{2,2}} = a_{11}b_{12} + b_{11}a_{21} + c_{11},$$

which has a length-5 expression, and Theorem 1 gives the matching lower bound.

For $K_{2,2}$,

$$P_{K_{2,2}} = a_{11}b_{12} + b_{11}a_{21} + c_{11} + b_{11}d_{11}b_{12}.$$

The expression

$$c_{11} + a_{11}b_{12} + b_{11}(a_{21} + d_{11}b_{12})$$

has length 7. Theorem 1 gives only the lower bound 6. If a length-6 expression existed, then each of the six labels would occur exactly once, so the expression would be read-once. The singleton monomial c_{11} must be separated by a sum gate. The remaining three monomials $a_{11}b_{12}$, $b_{11}a_{21}$, and $b_{11}d_{11}b_{12}$ admit no nontrivial read-once root decomposition. Indeed, any sum decomposition would have to partition the three monomials into two groups using disjoint sets of labels, but the first and third monomials share b_{12}, while the second and third share b_{11}. A product decomposition is also impossible: since there are three monomials, one factor would have to contribute a single monomial, which would force a common nonempty prefix or suffix among all three words. No such common prefix or suffix exists. Hence no read-once expression computes $P_{K_{2,2}}$, and the optimal length is 7. □

3 Generating Expressions for Directed Triangulated Grid Graphs by a Backtracking Method

The method is universal and suitable for generating expressions of any st-dag. An expression is derived by using intermediate subexpressions which are accumulated at the graph's vertices.

For TGGs, a subexpression accumulated at vertex (i, j) of $T_{m,n}$ corresponds to its subgraph located between vertices (i, j) and (m, n), and is denoted by $F(i, j)$. The subexpression accumulated at vertex $(1, 1)$ is the resulting expression. The method is implemented recursively using Algorithm 1.

The subexpression related to a single-vertex subgraph in Step 1 is accepted to be 1, so that its multiplication by another factor does not affect the result. Steps 2 and

Algorithm 1 Generating expressions for TGGs by the backtracking method

1: $F(m,n) \leftarrow 1$
2: $F(i,n) \leftarrow b_{i,n}b_{i+1,n}...b_{m-1,n}$ $(i < m)$
3: $F(m,j) \leftarrow a_{m,j}a_{m,j+1}...a_{m,n-1}$ $(j < n)$
4: $F(i,j) \leftarrow a_{i,j}F(i,j+1) + b_{i,j}F(i+1,j) + c_{i,j}F(i+1,j+1)$ $(i < m, j < n)$

3 describe expressions of subgraphs that are TGGs of size 1 or depth 1, respectively. The general case is processed in Step 4.

While the proposed algorithm optimizes the prefix parts of all subexpressions, the backtracking method can be applied by traversing the st-dag in the opposite direction. In this case suffix parts of subexpressions are optimized. The length of the derived expression will be the same.

3.1 Canonical transposition and algorithmic naturality

For TGGs there is a distinguished transposition map. This map is more than an abstract graph isomorphism: it is compatible with the recursive rules used to construct expressions. This compatibility is the reason why the equalities of lengths below are algorithmic equalities rather than merely consequences of the existence of some isomorphism.

Definition 1 For a directed triangulated grid graph $T_{m,n}$, the *canonical transposition*

$$\tau_{m,n} : T_{m,n} \longrightarrow T_{n,m}$$

is defined on vertices by

$$\tau_{m,n}(i,j) = (j,i).$$

On edge labels it acts by

$$a_{i,j} \mapsto b_{j,i}, \qquad b_{i,j} \mapsto a_{j,i}, \qquad c_{i,j} \mapsto c_{j,i}.$$

The same notation is used for the induced relabeling of formal expressions in the free noncommutative semiring.

Lemma 7 (Canonical transposition) The map $\tau_{m,n}$ is a source-target preserving isomorphism from $T_{m,n}$ onto $T_{n,m}$. It induces a bijection between the s-t paths of $T_{m,n}$ and the s-t paths of $T_{n,m}$, preserves label-occurrence length, and satisfies

$$\tau_{n,m} \circ \tau_{m,n} = \mathrm{id}.$$

Proof The vertex map sends $(1,1)$ to $(1,1)$ and (m,n) to (n,m). A horizontal edge $((i,j),(i,j+1))$ labeled $a_{i,j}$ is sent to the vertical edge $((j,i),(j+1,i))$ labeled $b_{j,i}$. A vertical edge $((i,j),(i+1,j))$ labeled $b_{i,j}$ is sent to the horizontal edge $((j,i),(j,i+1))$ labeled $a_{j,i}$. A diagonal edge $((i,j),(i+1,j+1))$ labeled $c_{i,j}$ is sent to the diagonal edge $((j,i),(j+1,i+1))$ labeled $c_{j,i}$. Thus directed edges, source, and target are preserved. The formula for the label map is involutive, so $\tau_{n,m} \circ \tau_{m,n} = \mathrm{id}$. Since relabeling changes only the names of labels and not their number of occurrences, expression length is preserved. □

Lemma 8 (Naturality of the backtracking recursion) Let $F_{m,n}(i,j)$ be the subexpression produced by Algorithm 1 for the subgraph of $T_{m,n}$ from (i,j) to (m,n). Then

$$\tau_{m,n}\big(F_{m,n}(i,j)\big) = F_{n,m}(j,i)$$

up to the order of summands. Consequently, the backtracking lengths satisfy

$$L_m(n) = L_n(m).$$

Proof The proof is by induction on the remaining distance $(m-i)+(n-j)$. The terminal case is immediate. If $i=m$ or $j=n$, the horizontal boundary product is carried by $\tau_{m,n}$ to the corresponding vertical boundary product in the transposed graph, and conversely. In the interior case,

$$F_{m,n}(i,j) = a_{i,j}F_{m,n}(i,j+1) + b_{i,j}F_{m,n}(i+1,j) + c_{i,j}F_{m,n}(i+1,j+1).$$

Applying $\tau_{m,n}$ and using the induction hypothesis gives

$$b_{j,i}F_{n,m}(j+1,i) + a_{j,i}F_{n,m}(j,i+1) + c_{j,i}F_{n,m}(j+1,i+1),$$

which is exactly the recursive expression for $F_{n,m}(j,i)$ after commuting the first two summands. Taking $(i,j)=(1,1)$ proves equality of the generated expressions up to relabeling and hence equality of their label-occurrence lengths. □

Proposition 9 *The length $L_m(n)$ of the expression $Ex(T_{m,n})$ derived by the backtracking method is defined recursively as follows:*

$$\begin{aligned} &L_m(1) = m-1, \quad L_1(n) = n-1 \\ &L_m(n) = L_m(n-1) + L_{m-1}(n) + L_{m-1}(n-1) + 3 \quad (m>1, n>1). \end{aligned} \tag{1}$$

Proof Initial statements (1) follow directly from Steps 1, 2, 3 of Algorithm 1. The resulting expression $Ex(T_{m,n})$ is equal to $a_{1,1}F(1,2)+b_{1,1}F(2,1)+c_{1,1}F(2,2)$. $F(1,2)$ is $Ex(T_{m,n-1})$, $F(2,1)$ is $Ex(T_{m-1,n})$ and $F(2,2)$ is $Ex(T_{m-1,n-1})$. Labels a_{11}, b_{11} and c_{11} are three additional labels in $Ex(T_{m,n})$. □

Lemma 10 Suppose $m>1$ and $n>1$. Then the length $L_m(n)$ of the expression $Ex(T_{m,n})$ derived by the backtracking method can be defined as follows:

$$L_m(n) = L_m(1) + \sum_{k=1}^{n} L_{m-1}(k) + \sum_{k=2}^{n-1} L_{m-1}(k) + 3(n-1).$$

Proof The proof is obtained by induction on n. By Proposition 9,

$$L_m(2) = L_m(1) + L_{m-1}(2) + L_{m-1}(1) + 3.$$

On the other hand,

$$\begin{aligned} &L_m(1) + \sum_{k=1}^{2} L_{m-1}(k) + \sum_{k=2}^{2-1} L_{m-1}(k) + 3(2-1) \\ =\ &L_m(1) + L_{m-1}(1) + L_{m-1}(2) + 3. \end{aligned}$$

Therefore, the lemma holds for $n = 2$.

The following is the induction step from $n-1$ to n, where we make use of Proposition 9.

$$\begin{aligned} L_m(n) &= L_m(n-1) + L_{m-1}(n) + L_{m-1}(n-1) + 3 \\ &= L_m(1) + \sum_{k=1}^{n-1} L_{m-1}(k) + \sum_{k=2}^{n-2} L_{m-1}(k) + 3(n-2) + L_{m-1}(n) + L_{m-1}(n-1) + 3 \\ &= L_m(1) + \sum_{k=1}^{n} L_{m-1}(k) + \sum_{k=2}^{n-1} L_{m-1}(k) + 3(n-1). \end{aligned}$$

□

Theorem 11 *The length $L_m(n)$ of the expression $Ex(T_{m,n})$ (m is considered as a constant) derived by the backtracking method is $O(n^m)$.*

Proof The proof is obtained by induction on m. By Proposition 9, $L_1(n) = n - 1 = O(n)$. Therefore, the theorem holds for $m = 1$. We will prove it for any $m > 1$, assuming it is correct for $m - 1$. According to Lemma 10, for $n > 1$

$$\begin{aligned} L_m(n) &= L_m(1) + \sum_{k=1}^{n} L_{m-1}(k) + \sum_{k=2}^{n-1} L_{m-1}(k) + 3(n-1) \\ &= \sum_{k=1}^{n} L_{m-1}(k) + \sum_{k=2}^{n-1} L_{m-1}(k) + 3(n-1) + m - 1. \end{aligned}$$

Hence, by the induction hypothesis, there exists a positive constant c such that

$$L_m(n) \le c\left(\sum_{k=1}^{n} k^{m-1} + \sum_{k=2}^{n-1} k^{m-1}\right) + 3(n-1) + m - 1.$$

In accordance with *Faulhaber's formula* for the sum of the p-th powers of the first n positive integers

$$\sum_{k=1}^{n} k^p = \frac{1}{p+1} \sum_{k=0}^{p} (-1)^k \binom{p+1}{k} B_k n^{p+1-k},$$

where B_k are *Bernoulli numbers* $\left(B_1 = -\frac{1}{2}\right)$. That is, $\sum_{k=1}^{n} k^p = O\left(n^{p+1}\right)$ and, therefore, $L_m(n) = O(n^m) + 3(n-1) + m - 1 = O(n^m)$. □

Specifically, for $n \ge 1$,

$$L_2(n) = n^2 + n - 1,\ L_3(n) = \frac{2}{3}n^3 + n^2 + \frac{4}{3}n - 1.$$

For example, $Ex(T_{3,4})$ derived by the backtracking method is

$a_{11}(a_{12}(a_{13}b_{14}b_{24} + b_{13}(a_{23}b_{24} + b_{23}a_{33} + c_{23}) + c_{13}b_{24}) +$
$b_{12}(a_{22}(a_{23}b_{24} + b_{23}a_{33} + c_{23}) + b_{22}a_{32}a_{33} + c_{22}a_{33}) + c_{12}(a_{23}b_{24} + b_{23}a_{33} + c_{23})) +$
$b_{11}(a_{21}(a_{22}(a_{23}b_{24} + b_{23}a_{33} + c_{23}) + b_{22}a_{32}a_{33} + c_{22}a_{33}) + b_{21}a_{31}a_{32}a_{33} + c_{21}a_{32}a_{33}) +$

$c_{11}(a_{22}(a_{23}b_{24} + b_{23}a_{33} + c_{23}) + b_{22}a_{32}a_{33} + c_{22}a_{33})$.

It contains 63 labels.

Thus, the length of $Ex(T_{m,n})$ obtained by the backtracking method grows polynomially with n. For large fixed values of m, however, the polynomial degree makes these expressions grow quickly. This behavior is illustrated by the data in Table 1. In addition, $L_m(n) = L_n(m)$. By Lemma 8, this equality is not based merely on the existence of an abstract isomorphism between $T_{m,n}$ and $T_{n,m}$. It follows from the canonical transposition $\tau_{m,n}$, which intertwines Algorithm 1 on $T_{m,n}$ with the same algorithm on $T_{n,m}$.

Table 1 Lengths of $Ex(T_{m,n})$ derived by the backtracking method.

$m \setminus n$	1	2	3	4	5	6	7	8	9	10
1	0	1	2	3	4	5	6	7	8	9
2	1	5	11	19	29	41	55	71	89	109
3	2	11	30	63	114	187	286	415	578	779
4	3	19	63	159	339	643	1119	1823	2819	4179
5	4	29	114	339	840	1825	3590	6535	11180	18181
6	5	41	187	643	1825	4493	9911	20039	37757	67121
7	6	55	286	1119	3590	9911	24318	54271	112070	216951
8	7	71	415	1823	6535	20039	54271	132863	299207	628231
9	8	89	578	2819	11180	37757	112070	299207	731280	1658721
10	9	109	779	4179	18181	67121	216951	628231	1658721	4048725

4 Generating Expressions for Directed Triangulated Grid Graphs by a Decomposition Method

The decomposition method recursively reveals smaller subgraphs inside a graph of regular structure. The resulting expression is obtained by a special composition of subexpressions describing these subgraphs. The existence of a decomposition method for a graph G is a sufficient condition for the existence of a polynomial-length expression for G, provided that in each recursive step the graph is split into a constant number of subgraphs whose sizes decrease proportionally.

The graph $T_{m,n}$ is conceptually split into left and right parts connected by *connecting edges* $a_{i,j}$ $(i = 1, 2, ..., m)$ and $c_{i,j}$ $(i = 1, 2, ..., m-1)$, where $a_{i,j}$ and $c_{i,j}$ are chosen so that sizes of subgraphs in the left and in the right parts are equal for even n and are distinguished by one for odd n. For example, these are edges $a_{13}, ..., a_{53}$, $c_{13}, ..., c_{43}$ in Figure 3(a)). Each connecting edge leaves a target of a subgraph belonging to the left part (its source is a source of the graph) and enters a source of a subgraph of the right part (its target is a target of the graph). Any path from the source to the target of the graph passes through one of the connecting edges. This decomposition is applied recursively to each revealed subgraph.

We denote by $F((p_1,p_2),(q_1,q_2))$ a subexpression of a subgraph with source (p_1,p_2) and target (q_1,q_2) (it has depth $q_1- p_1+1$ and size $q_2- p_2+1$). The decomposition procedure is initially invoked by substituting $(1,1)$ and (m,n) for (p_1,p_2) and (q_1,q_2), respectively and is implemented using Algorithm 2.

Algorithm 2 Generating expressions for TGGs by the decomposition method

1: $F((p_1,p_2),(p_1,p_2)) \leftarrow 1$

2: $F((p_1,p_2),(q_1,q_2)) \leftarrow b_{p_1,p_2}b_{p_1+1,p_2}...b_{q_1-1,p_2}\ (q_1>p_1, q_2=p_2)$

3: $F((p_1,p_2),(q_1,q_2)) \leftarrow F((p_1,p_2),(p_1,j))a_{p_1,j}F((p_1,j+1),(q_1,q_2))+$
$F((p_1,p_2),(p_1+1,j))a_{p_1+1,j}F((p_1+1,j+1),(q_1,q_2))+...+$
$F((p_1,p_2),(q_1,j))a_{q_1,j}F((q_1,j+1),(q_1,q_2))+$
$F((p_1,p_2),(p_1,j))c_{p_1,j}F((p_1+1,j+1),(q_1,q_2))+$
$F((p_1,p_2),(p_1+1,j))c_{p_1+1,j}F((p_1+2,j+1),(q_1,q_2))+...+$
$F((p_1,p_2),(q_1-1,j))c_{q_1-1,j}F((q_1,j+1),(q_1,q_2))\ (q_2>p_2)$
$j=\lceil\frac{q_2+p_2-1}{2}\rceil$ or $\lfloor\frac{q_2+p_2-1}{2}\rfloor$

The subexpression related to a single-vertex subgraph in Step 1 is accepted to be 1. Step 2 describes expressions of subgraphs that are TGGs of size 1. The general case is processed in Step 3.

There are m connecting edges labeled a and $m-1$ connecting edges labeled c, each of which connects two subgraphs. Thus, in the general case $Ex(T_{m,n})$ includes $4m-2$ subexpressions related to subgraphs of size approximately $n/2$ and $2m-1$ additional labels corresponding to connecting edges. This expression can be simplified by factoring out subexpressions which appear twice (e.g., from the left). After the simplification, Step 3 of Algorithm 2 takes the form:

$$
\begin{aligned}
&F((p_1,p_2),(q_1,q_2)) \leftarrow \\
&F((p_1,p_2),(p_1,j))\,(a_{p_1,j}F((p_1,j+1),(q_1,q_2))+c_{p_1,j}F((p_1+1,j+1),(q_1,q_2)))+\\
&F((p_1,p_2),(p_1+1,j))\,(a_{p_1+1,j}F((p_1+1,j+1),(q_1,q_2))+c_{p_1+1,j}F((p_1+2,j+1),(q_1,q_2)))\\
&+...+\\
&F((p_1,p_2),(q_1-1,j))\,(a_{q_1-1,j}F((q_1-1,j+1),(q_1,q_2))+c_{q_1-1,j}F((q_1,j+1),(q_1,q_2)))\\
&+F((p_1,p_2),(q_1,j))a_{q_1,j}F((q_1,j+1),(q_1,q_2)).
\end{aligned}
\tag{2}
$$

The number of recursive subexpressions corresponding to subgraphs of size approximately $n/2$ is thereby reduced to $3m-1$. To factor out the longer repeated subexpression, we choose $j=\lceil\frac{q_2+p_2-1}{2}\rceil$.

The lengths of the expressions $Ex(T_{m,n})$ $(m=1,...,10;\ n=1,...,10)$ derived by the decomposition method based on Algorithm 2 with Step 3 implemented by (2) are presented in Table 2.

The values in Table 2 are substantially smaller than the corresponding values in Table 1 for the displayed range; the asymptotic improvement follows from Theorem 13. However, for $n=2$, the backtracking method gives shorter representations than the basic decomposition method.

Table 2 Lengths of $Ex(T_{m,n})$ derived by the decomposition method.

$m \setminus n$	1	2	3	4	5	6	7	8	9	10
1	0	1	2	3	4	5	6	7	8	9
2	1	5	10	16	22	29	36	44	51	59
3	2	12	27	47	68	95	122	156	184	219
4	3	22	56	105	160	236	312	415	498	609
5	4	35	100	199	319	494	669	920	1123	1409
6	5	51	162	338	569	920	1271	1797	2231	2868
7	6	70	245	531	937	1574	2211	3199	4039	5313
8	7	92	352	787	1453	2525	3597	5306	6812	9158
9	8	117	486	1115	2150	3851	5552	8325	10866	14913
10	9	145	650	1524	3064	5639	8214	12490	16571	23193

For $m \geq 3$, the expressions $Ex(T_{m,2})$ and $Ex(T_{m,3})$ can be further simplified (the *improved decomposition method*). Specifically, when decomposing $T_{m,2}$ or $T_{m,3}$, subgraphs of size 1 are revealed on both sides or on one side, respectively, of the decomposed graph. Expressions of these subgraphs are label products (see Step 2 of Algorithm 2), and their common labels can also be factored out.

For example, $Ex(T_{5,2})$ derived by the decomposition method is

$$a_{11}b_{12}b_{22}b_{32}b_{42} + c_{11}b_{22}b_{32}b_{42} + b_{11}(a_{21}b_{22}b_{32}b_{42} + c_{21}b_{32}b_{42}) + \\ b_{11}b_{21}(a_{31}b_{32}b_{42} + c_{31}b_{42}) + b_{11}b_{21}b_{31}(a_{41}b_{42} + c_{41}) + b_{11}b_{21}b_{31}b_{41}a_{51}. \tag{3}$$

It contains 35 labels. After the transformations described above, this expression will look like this:

$$(a_{11}b_{12} + c_{11})b_{22}b_{32}b_{42} + b_{11}((a_{21}b_{22} + c_{21})b_{32}b_{42} + \\ b_{21}((a_{31}b_{32} + c_{31})b_{42} + b_{31}(a_{41}b_{42} + c_{41} + b_{41}a_{51}))). \tag{4}$$

In general, the number of labels in $Ex(T_{m,2})$ after such simplification decreases by $2(1+2+...+m-2) = (m-1)(m-2)$. In particular, the length of $Ex(T_{5,2})$ decreases by 12, and the expression (4) contains 23 labels.

As noted above, when decomposing $T_{m,3}$, subgraphs of size 1 are revealed only on one side. For example, $Ex(T_{5,3})$ derived by the decomposition method is

$$a_{11}(a_{12}b_{13}b_{23}b_{33}b_{43} + c_{12}b_{23}b_{33}b_{43}) + (a_{11}b_{12} + c_{11} + b_{11}a_{21})(a_{22}b_{23}b_{33}b_{43} + c_{22}b_{33}b_{43}) + \\ F((1,1),(3,2))(a_{32}b_{33}b_{43} + c_{32}b_{43}) + F((1,1),(4,2))(a_{42}b_{43} + c_{42}) + F((1,1),(5,2))a_{52},$$

where subexpressions of three left subgraphs of size 2 (depths 3, 4, 5) are denoted for brevity as in Algorithm 2. In accordance with Table 2, this expression contains 100 labels. It can be shortened by a two-stage simplification (first, factoring out common right subproducts in local sums; second, factoring out common labels b across the

entire expression) as follows:

$$((a_{11}(a_{12}b_{13}+c_{12})b_{23}+(a_{11}b_{12}+c_{11}+b_{11}a_{21})(a_{22}b_{23}+c_{22}))b_{33}+$$
$$F((1,1),(3,2))(a_{32}b_{33}+c_{32}))b_{43}+F((1,1),(4,2))(a_{42}b_{43}+c_{42})+F((1,1),(5,2))a_{52}.$$

Compared to Table 2, the lengths of $F((1,1),(3,2))$, $F((1,1),(4,2))$, $F((1,1),(5,2))$ are reduced by $(m-1)(m-2)$, i.e., by 2, 6, 12 and are equal to 10, 16, 23, respectively. Ultimately, this expression contains 71 labels.

In general, the above two-stage simplification of $Ex(T_{m,3})$ reduces its length by $1+2+...+m-2=\frac{(m-1)(m-2)}{2}$ at the first stage and by $1+2+...+m-3=\frac{(m-2)(m-3)}{2}$ on the second stage. In total the length decreases by $(m-2)^2$. In particular, in the previous example, along with internal reductions in $F((1,1),(3,2))$, $F((1,1),(4,2))$, $F((1,1),(5,2))$, the length of $Ex(T_{5,3})$ further decreases by 9.

Proposition 12 *The length $L_m(n)$ of the expression $Ex(T_{m,n})$ derived by the improved decomposition method is defined recursively as follows:*

$$L_m(1) = m-1 \tag{5}$$
$$L_m(2) = \frac{1}{2}m^2+\frac{5}{2}m-2 \quad (m\geq 1) \tag{6}$$
$$L_m(3) = \frac{1}{6}m^3+\frac{3}{2}m^2+\frac{10}{3}m-4 \quad (m\geq 2) \tag{7}$$
$$L_m(n) = 2\sum_{k=1}^{m-1}L_k\left(\left\lfloor\frac{n}{2}\right\rfloor\right)+L_m\left(\left\lfloor\frac{n}{2}\right\rfloor\right)+\sum_{k=1}^{m}L_k\left(\left\lceil\frac{n}{2}\right\rceil\right)+2m-1 \tag{8}$$
for $n>3$, and also for $m=1, n=3$.

Proof The initial statement 5 follows directly from Step 2 of Algorithm 2.

The general formula (8) is based on the structure of the statement (2). For each subgraph of depth $m_1=1,\ldots,m$ on the left, there exists a complementary subgraph of depth $m_2=m-m_1+1$ on the right. Since the split is located at $\left\lceil\frac{q_2+p_2-1}{2}\right\rceil$, there are $2m-1$ subexpressions related to subgraphs of size $\left\lfloor\frac{n}{2}\right\rfloor$, and m subexpressions related to subgraphs of size $\left\lceil\frac{n}{2}\right\rceil$. In addition, as previously noted, there are $2m-1$ additional labels. Thus, before the additional size-2 and size-3 simplifications, the same decomposition count holds for every $n>1$. In the improved method, (8) is used as the recursive clause for $n>3$ and also for the harmless case $m=1, n=3$, where no additional simplification is available.

However, as outlined earlier, for $m\geq 3$, the number of labels in $Ex(T_{m,2})$ and $Ex(T_{m,3})$ can be decreased further by $(m-1)(m-2)$ and $(m-2)^2$, respectively. Therefore, combining the initial statement (5), the general formula (8), and applying Faulhaber's formula, we obtain

$$\begin{aligned}L_m(2) &= 2\sum_{k=1}^{m-1}L_k(1)+L_m(1)+\sum_{k=1}^{m}L_k(1)+2m-1-(m-1)(m-2)\\ &= 3\sum_{k=1}^{m-1}L_k(1)+2L_m(1)-m^2+5m-3\end{aligned}$$

$$= 3\sum_{k=1}^{m-1}(k-1) + 2(m-1) - m^2 + 5m - 3$$
$$= 3\frac{(m-2)(m-1)}{2} + 2m - 2 - m^2 + 5m - 3$$
$$= \frac{1}{2}m^2 + \frac{5}{2}m - 2$$

and

$$\begin{aligned} L_m(3) &= 2\sum_{k=1}^{m-1} L_k(1) + L_m(1) + \sum_{k=1}^{m} L_k(2) + 2m - 1 - (m-2)^2 \\ &= 2\sum_{k=1}^{m-1}(k-1) + m - 1 + \sum_{k=1}^{m}\left(\frac{1}{2}k^2 + \frac{5}{2}k - 2\right) - m^2 + 6m - 5 \\ &= 2\frac{(m-2)(m-1)}{2} + m - 1 + \frac{1}{6}m^3 + \frac{1}{4}m^2 + \frac{1}{12}m + \frac{5}{4}m(m+1) - 2m \\ &\quad -m^2 + 6m - 5 \\ &= \frac{1}{6}m^3 + \frac{3}{2}m^2 + \frac{10}{3}m - 4. \end{aligned}$$

For $m = 2$ and for $m = 1$, $(m-1)(m-2)$ equals 0, and, for this reason, (6) holds already for $m \geq 1$. Also, $(m-2)^2$ equals 0 for $m = 2$, and, for this reason, (7) holds already for $m \geq 2$. This completes the proof. □

Theorem 13 *The length $L_m(n)$ of the expression $Ex(T_{m,n})$ (m is considered as a constant) derived by both the decomposition and the improved decomposition methods is $O\left(n\log^{m-1} n\right)$.*

Proof For each fixed m, define

$$\widehat{L}_m(n) = \max_{1\leq r\leq n} L_m(r).$$

It is enough to prove $\widehat{L}_m(n) = O(n\log^{m-1} n)$, since $L_m(n) \leq \widehat{L}_m(n)$. For $m = 1$, $\widehat{L}_1(n) = n - 1 = O(n)$. Assume the statement is known for $1, \ldots, m-1$. The exceptional values $r \leq 3$ in Proposition 12 are bounded by a constant depending only on m. For $r > 3$, (8) and the inequalities

$$\left\lfloor \frac{r}{2} \right\rfloor \leq \left\lceil \frac{n}{2} \right\rceil, \qquad \left\lceil \frac{r}{2} \right\rceil \leq \left\lceil \frac{n}{2} \right\rceil \quad (r \leq n)$$

give, after taking the maximum over $1 \leq r \leq n$,

$$\widehat{L}_m(n) \leq 2\widehat{L}_m\left(\left\lceil \frac{n}{2} \right\rceil\right) + 3\sum_{k=1}^{m-1} \widehat{L}_k\left(\left\lceil \frac{n}{2} \right\rceil\right) + C_m,$$

where C_m is a constant. By the induction hypothesis,

$$\sum_{k=1}^{m-1} \widehat{L}_k\left(\left\lceil \frac{n}{2} \right\rceil\right) = O(n\log^{m-2} n).$$

Therefore

$$\widehat{L}_m(n) \leq 2\widehat{L}_m\left(\left\lceil \frac{n}{2} \right\rceil\right) + O(n\log^{m-2} n).$$

The master theorem gives $\widehat{L}_m(n) = O(n \log^{m-1} n)$, and hence $L_m(n) = O(n \log^{m-1} n)$. The basic decomposition method differs from the improved method only in the constant-size boundary simplifications for $n = 2$ and $n = 3$; removing those simplifications changes lower-order constants but not this asymptotic bound. □

Corollary 14 (global optimality for TGGs of depths at most four) For $m \in \{1, 2, 3, 4\}$,

$$\ell(P_{T_{m,n}}) = \Theta_m \left(n \log^{m-1} n \right).$$

In particular, for these depths the decomposition and improved decomposition methods are asymptotically optimal among all expressions over the free noncommutative semiring.

Proof The lower bound is Theorem 5. The upper bound is Theorem 13. □

For later reference we also name the variant used in the last part of this section. The *alternating decomposition method* applies the same separator-and-factoring idea recursively, but splits each current subgraph through its longer side: vertically when the horizontal size is at least the depth, and horizontally after applying the canonical transposition when the depth is larger. Thus the method is defined before it is used in the interval-decomposition model below.

4.1 All-depth optimality in the balanced interval-decomposition model

The preceding corollary is an unrestricted formula-size statement only for small fixed depths. We next record a complementary all-depth result for the recursive model used by the decomposition algorithms. To avoid hiding the restriction in informal language, we formulate this model as a recursive grammar.

Definition 2 (balanced column-interval decomposition formula) Fix the depth m. A balanced column-interval decomposition formula for a TGG subproblem is generated by the following recursive grammar.

1. The leaves are the base path polynomials of subgraphs whose horizontal size is at most a constant $N_0(m)$, and the one-column vertical products.
2. Every non-base internal node chooses a vertical separator so that the left and right horizontal sizes differ by at most a constant depending only on m. The node expands the path polynomial by partitioning paths according to the unique horizontal or diagonal edge by which they cross the separator.
3. For each crossing class, the grammar creates the corresponding left interval subproblem, the crossing label, and the corresponding right interval subproblem. Empty factors and unit factors are deleted. Ordinary local factoring by distributivity is allowed, but only after the crossing-class expansion has created these recursive occurrences.
4. The representation is a formula tree. Thus two equal-looking recursive subproblems that occur in two different crossing classes are counted as two occurrences; no sharing as a syntax DAG is allowed.

The basic, improved, and alternating decomposition algorithms all belong to this model, after applying canonical transposition to the subproblems that the alternating rule splits horizontally.

Let $I_m(n)$ be the minimum length of a balanced column-interval decomposition formula for $P_{T_{m,n}}$.

Lemma 15 (separator-occurrence lemma) For every fixed $m > 1$ there is a constant $N_0(m)$ such that every non-base separator expansion of a depth-m balanced interval formula on an interval of horizontal size $N > N_0(m)$ has the following property. If the separator produces left and right intervals of sizes N_L and N_R, then, for each $k = 1, 2, \ldots, m-1$, the expansion contains at least one recursive occurrence of a depth-k left interval subproblem of horizontal size $N_L + O_m(1)$ and at least one recursive occurrence of a depth-k right interval subproblem of horizontal size $N_R + O_m(1)$, apart from unit factors and boundary terms whose total contribution is $O_m(1)$. After increasing the base threshold $N_0(m)$, if necessary, the constant endpoint shifts in horizontal sizes are absorbed into the $O_m(1)$ term; in the recurrence below we therefore write the two child sizes simply as N_L and N_R. Consequently

$$I_m(N) \geq I_m(N_L) + I_m(N_R) + \sum_{k=1}^{m-1} \big(I_k(N_L) + I_k(N_R)\big) - O_m(1).$$

Proof A path crossing a vertical separator in a TGG uses exactly one of the m horizontal crossing edges or one of the $m-1$ diagonal crossing edges. For a horizontal crossing edge in row r, the left factor is a TGG interval subproblem of depth r and the right factor is a TGG interval subproblem of depth $m-r+1$. For a diagonal crossing edge from row r to row $r+1$, the left factor has depth r and the right factor has depth $m-r$. Thus the horizontal crossing classes already provide left depths $1, \ldots, m$ and right depths $m, \ldots, 1$, while the diagonal crossing classes provide left and right depths $1, \ldots, m-1$. In particular, for every $k < m$, at least one depth-k subproblem appears on the left side and at least one depth-k subproblem appears on the right side.

Definition 2 requires the separator expansion to create these left and right recursive occurrences before local distributive factoring is applied. Such factoring may move a common factor outside a sum, but it cannot identify two occurrences as one shared object, because the model is a formula tree. Moreover, different crossing classes carry different crossing labels, so deleting an occurrence would delete all monomials using that crossing label. The only exceptions are unit factors, empty factors, and a bounded number of boundary cases depending only on m; these are absorbed into the $O_m(1)$ term. This gives the displayed recurrence. □

Theorem 16 (all-depth optimality in the balanced interval model) *For every fixed $m \geq 1$,*

$$I_m(n) = \Theta_m\left(n \log^{m-1} n\right).$$

Proof The upper bound follows from Theorem 13. We prove the lower bound by induction on m. For $m = 1$, the graph has one source-target path of length $n-1$, so $I_1(n) = \Omega(n)$.

Let $m > 1$ and assume the statement for all smaller depths. Consider a non-base formula occurrence of depth m and horizontal size N. By Definition 2, its first separator produces intervals N_L and N_R satisfying

$$N_L, N_R \le \lceil N/2 \rceil + O_m(1), \qquad N_L + N_R = N + O_m(1).$$

Lemma 15 and the induction hypothesis give

$$\begin{aligned} I_m(N) &\ge I_m(N_L) + I_m(N_R) + \Omega_m\big((N_L + N_R)\log^{m-2}(N+2)\big) \\ &= I_m(N_L) + I_m(N_R) + \Omega_m\big(N\log^{m-2}(N+2)\big). \end{aligned}$$

where the equality uses $N_L + N_R = N + O_m(1)$ and the balanced choice of the separator. Solving this balanced lower recurrence gives $I_m(N) = \Omega_m(N\log^{m-1} N)$. Indeed, for sufficiently small $A_m > 0$, assume inductively that $I_m(s) \ge A_m s\log^{m-1}(s+2)$ for all $s < N$. Since the split is balanced,

$$N\log^{m-1}(N+2) - N_L\log^{m-1}(N_L+2) - N_R\log^{m-1}(N_R+2) \le C_m N\log^{m-2}(N+2)$$

for a constant C_m depending only on m. This follows from the mean-value theorem applied to $x\log^{m-1}(x+2)$ on the two balanced intervals. Choosing A_m small relative to the implicit constant in the separator term and to C_m closes the induction. The finitely many smaller values of N are absorbed into the constant. Thus $I_m(n) = \Omega_m(n\log^{m-1} n)$, and the upper bound completes the proof. □

Theorem 16 proves all-depth optimality for the precise balanced recursive decomposition paradigm of Definition 2. It does not assert an unrestricted formula lower bound for all depths; the unrestricted all-depth problem is the binomial-language route isolated in Conjecture 1. The stronger unrestricted statement $\ell(P_{T_{m,n}}) = \Omega_m(n\log^{m-1} n)$ for every fixed m remains a global formula-size question; Conjecture 1 identifies one precise route to it through binomial-language lower bounds.

Specifically, for n that is a power of two ($n = 2^p$ for some positive integer $p \ge 1$) we obtain the following explicit formulas for the improved decomposition method:

$$L_2(n) = \frac{3}{2}n\log_2 n + n, \quad L_3(n) = \frac{9}{8}n\log_2^2 n + \frac{15}{8}n\log_2 n + 3n - 2.$$

For example, $Ex(T_{3,4})$ derived by the improved decomposition method is

$$\begin{aligned} &a_{11}(a_{12}((a_{13}b_{14} + c_{13})b_{24} + b_{13}(a_{23}b_{24} + c_{23} + b_{23}a_{33})) + c_{12}(a_{23}b_{24} + c_{23} + b_{23}a_{33})) + \\ &(a_{11}b_{12} + c_{11} + b_{11}a_{21})(a_{22}(a_{23}b_{24} + c_{23} + b_{23}a_{33}) + c_{22}a_{33}) + \\ &((a_{11}b_{12} + c_{11} + b_{11}a_{21})b_{22} + b_{11}(c_{21} + b_{21}a_{31}))a_{32}a_{33}. \end{aligned}$$

It contains 43 labels.

Table 3 presenting the lengths of the expressions $Ex(T_{m,n})$ ($m = 1, ..., 10$; $n = 1, ..., 10$) derived by the improved decomposition method shows the numerical advantage of this method over the decomposition one.

The vertical decomposition method always splits by the column coordinate. Consequently, unlike the backtracking recurrence, it is not invariant under interchanging m and n. The numerical data in Tables 2 and 3 illustrate this asymmetry and show that treating the longer side as the size parameter often gives a shorter expression.

Table 3 Lengths of $Ex(T_{m,n})$ derived by the improved decomposition method.

$m \setminus n$	1	2	3	4	5	6	7	8	9	10
1	0	1	2	3	4	5	6	7	8	9
2	1	5	10	16	22	29	36	44	51	59
3	2	10	24	43	63	89	115	148	175	209
4	3	16	44	87	135	203	272	367	442	544
5	4	23	71	151	247	391	540	758	929	1175
6	5	31	106	238	409	676	957	1387	1729	2242
7	6	40	150	351	632	1084	1566	2329	2952	3917
8	7	50	204	493	928	1644	2415	3668	4726	6407
9	8	61	269	667	1310	2388	3557	5497	7198	9957
10	9	73	346	876	1792	3351	5050	7918	10535	14853

This motivates the alternating decomposition method defined above: whenever a current subgraph has depth greater than its size, we decompose it horizontally into upper and lower parts by applying the canonical transposition and the same separator rule.

For example, as noted above, expressions $Ex(T_{5,2})$ derived by the decomposition method (3) and the improved decomposition method (4) contain 35 and 23 labels, respectively. Using the alternating decomposition method for $T_{5,2}$ we obtain the following expression which contains only 22 labels:

$$b_{11}b_{21}(b_{31}(b_{41}a_{51} + c_{41} + a_{41}b_{42}) + c_{31}b_{42}) +$$
$$(b_{11}(b_{21}a_{31} + c_{21}) + (b_{11}a_{21} + c_{11} + a_{11}b_{12})b_{22})b_{32}b_{42}.$$

The lengths of the expressions $Ex(T_{m,n})$ $(m = 1, ..., 10;\ n = 1, ..., 10)$ derived by the alternating decomposition method are presented in Table 4. The expression $Ex(T_{3,3})$ is obtained by the improved decomposition method.

Table 4 Lengths of $Ex(T_{m,n})$ derived by the alternating decomposition method.

$m \setminus n$	1	2	3	4	5	6	7	8	9	10
1	0	1	2	3	4	5	6	7	8	9
2	1	5	10	16	22	29	36	44	51	59
3	2	10	24	43	63	89	115	148	175	209
4	3	16	43	87	134	201	271	367	441	542
5	4	22	63	134	237	372	513	724	901	1152
6	5	29	89	201	372	615	868	1262	1610	2135
7	6	36	115	271	513	868	1345	2007	2597	3535
8	7	44	148	367	724	1262	2007	3014	3961	5498
9	8	51	175	441	901	1610	2597	3961	5672	8026
10	9	59	209	542	1152	2135	3535	5498	8026	11233

In Table 4 and in the following paragraph, $L_m(n)$ denotes the length obtained by the alternating decomposition method. The rule "split through the longer side" is canonical with respect to the transposition $\tau_{m,n}$. After applying $\tau_{m,n}$, a vertical split becomes the corresponding horizontal split of the transposed subgraph, and a horizontal split becomes the corresponding vertical split. If the two dimensions are equal, the two possible choices are paired by the involution $\tau_{m,m}$. Hence the recursive decomposition trees for $T_{m,n}$ and $T_{n,m}$ are paired node by node; paired nodes contribute the same number of label occurrences after relabeling. Consequently,

$$L_m(n) = L_n(m)$$

for the alternating decomposition method. This equality is therefore a naturality property of the rule with respect to canonical transposition, not a consequence of graph isomorphism alone. Table 4 shows that, for the displayed values, the alternating rule gives expressions no longer than those obtained by the improved decomposition method.

For fixed m, the alternating decomposition has the same asymptotic upper bound as the improved decomposition method. Along any recursive branch, as long as the horizontal size of a subproblem is at least its depth, the vertical split is governed by the recurrence used in Theorem 13. If a horizontal split is used, then the horizontal size is smaller than the depth, and both dimensions of the subproblem are at most m; all subsequent subproblems have bounded size depending only on m. These bounded subproblems can be absorbed into the constants in the proof of Theorem 13. Consequently, the length of $Ex(T_{m,n})$ derived by the alternating decomposition method is also $O\left(n \log^{m-1} n\right)$.

Thus, given a graph $T_{m,n}$ of any depth m, the length of $Ex(T_{m,n})$ derived by the decomposition method and its variations grows quasi-linearly with the size n of the graph.

As shown in [28], the lengths of expressions $Ex(G_{m,n})$ derived by the backtracking and decomposition methods are also $O\left(n^m\right)$ and $O\left(n \log^{m-1} n\right)$, respectively. In other words, adding diagonal edges labeled c to a directed grid graph only increases lengths of corresponding expressions numerically but does not change their asymptotic order.

5 Generating Expressions for Directed King Graphs by a Backtracking Method

A subgraph enclosed between vertices (i, j) and (m, n) of $G_{m,n}$ or $T_{m,n}$ is also a grid graph or a TGG, respectively. The addition of diagonal edges labeled c results in new paths, which essentially duplicate existing ones in $G_{m,n}$, but provide shorter routes to the right and downward. However, edges labeled d, although they ultimately lead to the target, also lead upwards in certain areas. For this reason, a subgraph of $K_{m,n}$ positioned between source (i, j) and target (m, n) is generally a pentagon (if $i < m$ and $i - 1 < n - j$). In the special cases, this subgraph reduces to a trapezoid (if $i = m, i-1 < n-j$ or $i < m, i-1 \geq n-j$) or to a triangle (if $i = m$ and $i-1 \geq n-j$).

This subgraph (see Fig. 4) is a king graph with a cut upper-left corner and will be called an *upper-cut king graph*. We denote such a subgraph by $K^h_{m,n}$, where m and n are the depth and the size of the initial rectangular king graph (with the same meaning for $K^h_{m,n}$), and h is the number of vertices in each side of the cut corner. For example, the graph in Fig. 4 will be denoted by $K^2_{5,6}$. An uncut king graph of depth m and size n may be denoted by $K^0_{m,n}$.

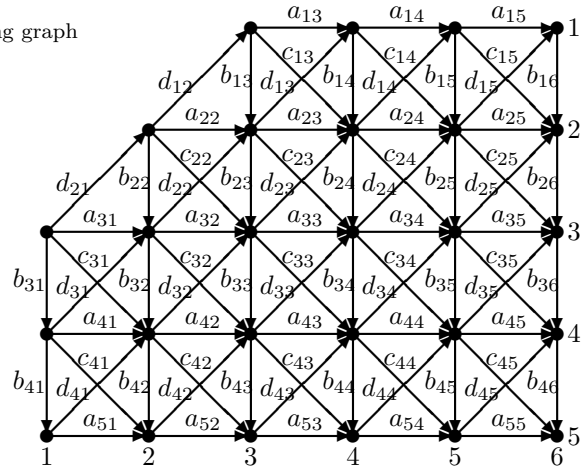


Fig. 4 An example of an upper-cut king graph

In order to apply a backtracking method to a king graph, we denote by $F(i,j)$ a subexpression accumulated at vertex (i,j) of $K_{m,n}$ corresponding to its subgraph which is positioned between vertices (i,j) and (m,n). The subexpression accumulated at vertex $(1,1)$ is the resulting expression. The method is implemented recursively using Algorithm 3.

Algorithm 3 Generating expressions for directed king graphs by the backtracking method

1: **if** $m = 1$ and $n > 1$ **then**
2: $\quad F(1,1) \leftarrow a_{11}a_{1,2}...a_{1,n-1}$
3: **else**
4: $\quad F(m,n) \leftarrow 1$
5: $\quad F(i,n) \leftarrow b_{i,n}b_{i+1,n}...b_{m-1,n}$ $(i < m)$
6: $\quad F(m,j) \leftarrow a_{m,j}F(m,j+1) + d_{m-1,j}F(m-1,j+1)$ $(j < n)$
7: $\quad F(1,j) \leftarrow a_{1,j}F(1,j+1) + b_{1,j}F(2,j) + c_{1,j}F(2,j+1)$ $(j < n)$
8: $\quad F(i,j) \leftarrow a_{i,j}F(i,j+1) + b_{i,j}F(i+1,j) + c_{i,j}F(i+1,j+1) + d_{i-1,j}F(i-1,j+1)$ $(1 < i < m, j < n)$

The special case for a non-single-vertex graph of depth 1 is described separately in Steps 1 - 2. In all other cases (a graph depth is greater than 1), even when the source of the subgraph is on the upper or lower rows of the graph, moving along this row is not the only way to reach the target.

The subexpression related to a single-vertex subgraph in Step 4 is accepted to be 1. Steps 5, 6, 7 present expressions of subgraphs whose sources are located on the right column, the lower row and the upper row of the initial king graph, respectively. The general case, where a graph traversal can be continued from a given vertex in four directions, is processed in Step 8 and includes, among other things, a situation where a subgraph source is on the left column of the initial graph.

Proposition 17 *Denote by $L^h_m(n)$ the length of the expression $Ex(K^h_{m,n})$ derived by the backtracking method. Then*

$$L^1_2(1) = 0, \quad L^0_2(1) = 1 \tag{9}$$

$$L^1_2(n) = L^1_2(n-1) + L^0_2(n-1) + 2 \quad (n > 1) \tag{10}$$

$$L^0_2(n) = 2\left(L^0_2(n-1) + L^1_2(n-1)\right) + 5 \quad (n > 1) \tag{11}$$

Proof The initial statements (9) follow directly from Steps 4 and 5 of Algorithm 3, respectively. The statement (10) follows from Step 6.

In accordance with Step 7,

$$\begin{aligned} L^0_2(n) &= L^0_2(n-1) + L^1_2(n) + L^1_2(n-1) + 3 \\ &= L^0_2(n-1) + L^1_2(n-1) + L^0_2(n-1) + L^1_2(n-1) + 2 + 3 \\ &= 2\left(L^0_2(n-1) + L^1_2(n-1)\right) + 5. \end{aligned}$$

□

Theorem 18 *Denote by $L^h_m(n)$ the length of the expression $Ex(K^h_{m,n})$ derived by the backtracking method. Then*

$$L^0_2(n) = 3^n - 2, \quad L^1_2(n) = \frac{3^n}{2} - \frac{3}{2}.$$

Proof The proof is obtained by induction on n. By Proposition 17, $L^0_2(1) = 1 = 3 - 2$ and $L^1_2(1) = 0 = \frac{3}{2} - \frac{3}{2}$. Therefore, the theorem holds for $n = 1$. We will prove it for any $n > 1$, assuming it is correct for $n - 1$.

By the statement (11) of Proposition 17,

$$\begin{aligned} L^0_2(n) &= 2\left(L^0_2(n-1) + L^1_2(n-1)\right) + 5 \\ &= 2\left(3^{n-1} - 2 + \frac{3^{n-1}}{2} - \frac{3}{2}\right) + 5 = 3 \cdot 3^{n-1} - 4 - 3 + 5 = 3^n - 2. \end{aligned}$$

By the statement (10) of Proposition 17,

$$L^1_2(n) = L^1_2(n-1) + L^0_2(n-1) + 2 = \frac{3^{n-1}}{2} - \frac{3}{2} + 3^{n-1} - 2 + 2 = \frac{3^n}{2} - \frac{3}{2}.$$

□

For example, $Ex(K_{2,3})$ derived by the backtracking method is

$$a_{11}(a_{12}b_{13} + b_{12}(a_{22} + d_{12}b_{13}) + c_{12}) +$$
$$b_{11}(a_{21}(a_{22} + d_{12}b_{13}) + d_{11}(a_{12}b_{13} + b_{12}(a_{22} + d_{12}b_{13}) + c_{12})) + c_{11}(a_{22} + d_{12}b_{13}).$$

It contains 25 labels.

Thus, the backtracking method applied to a king graph of size n produces expressions of exponential length in n, already for graphs of depth 2. Moreover, this exponential lower bound persists for every depth $m \geq 2$. For $m = 2$ this is exactly Theorem 18. Let $m > 2$, and let D_r and U_r be the lengths of the subexpressions accumulated by Algorithm 3 at vertices $(m, n-r+1)$ and $(m-1, n-r+1)$, respectively, where $1 \leq r \leq n$. Then $D_1 = 0$ and $U_1 = 1$. Steps 6 and 8 give, for $r > 1$,

$$D_r = D_{r-1} + U_{r-1} + 2,$$
$$U_r = U_{r-1} + D_r + D_{r-1} + W_{r-1} + 4 \geq U_{r-1} + D_r + D_{r-1} + 3,$$

where W_{r-1} is the length of the subexpression accumulated at $(m-2, n-r+2)$ and is nonnegative. Comparing these inequalities with (10) and (11) shows by induction on r that $D_r \geq L_2^1(r)$ and $U_r \geq L_2^0(r)$. In addition, $F(m-1, 1)$ occurs as an actual subexpression of $F(1, 1)$ through the chain of terms $b_{1,1}F(2, 1), b_{2,1}F(3, 1), \ldots, b_{m-2,1}F(m-1, 1)$ in Algorithm 3. Therefore,

$$L_m^0(n) \geq U_n \geq L_2^0(n) = 3^n - 2 \qquad (m \geq 2).$$

Hence, for all fixed depths $m \geq 2$, the backtracking method yields expressions of at least exponential length in n.

6 Generating Expressions for Directed King Graphs by a Decomposition Method

We apply the decomposition method to a king graph in a similar manner as for a TGG in Section 4. $K_{m,n}$ is conceptually split into left and right parts connected by *connecting edges* $a_{i,j}$ $(i = 1, 2, ..., m)$, $c_{i,j}$ $(i = 1, 2, ..., m-1)$ and $d_{i,j}$ $(i = 1, 2, ..., m-1)$ so that j is chosen in the middle of a row (edges $a_{13}, ..., a_{53}$, $c_{13}, ..., c_{43}$, $d_{13}, ..., d_{43}$ in Figure 3(b)). The graph is decomposed into $2m$ subgraphs of size approximately $n/2$. The left upper subgraph is connected with two right subgraphs (via edges a_{13} and c_{13} in Figure 3(b)), the left lower subgraph is connected with two right subgraphs as well (via edges a_{53} and d_{43} in Figure 3(b)), and each of the other left subgraphs is connected with three right subgraphs by edges labeled a, c, d.

In a king graph, targets of left subgraphs can be reached not only from above, but also from below. For this reason, subgraphs revealed from the left except for the lower one are generally pentagons. Each such subgraph (see Figure 5(a)) is a king graph with a cut lower-right corner and will be called an *under-cut king graph*. We denote such a subgraph by $K_{m,n}^{-h}$, where m and n are the depth and the size of the initial rectangular king graph (with the same meaning for $K_{m,n}^{-h}$) and h is the number

of vertices in each side of the cut corner. For example, the graph in Figure 5(a) will be denoted by $K_{5,4}^{-2}$. It should also be noted that there are paths leading upward from sources of right subgraphs. Thus, these subgraphs, except for the upper one, are upper-cut king graphs (Figure 4).

Upper-cut and under-cut subgraphs are decomposed in the same way. New king subgraphs and upper-cut or under-cut subgraphs are revealed in the course of decomposition. In addition, king subgraphs with both the upper-left and the lower-right corners cut off may appear (see Figure 5(b)). We call such subgraphs, which are hexagonal in the general case, *two-cut king graphs* and denote them by $K_{m,n}^{h_1,-h_2}$, where m and n are the depth and the size of the initial rectangular king graph (with the same meaning for $K_{m,n}^{h_1,-h_2}$) and h_1 and h_2 are the numbers of vertices in each side of the upper and the lower cut corners, respectively. For example, the graph in Figure 5(b) will be denoted by $K_{5,4}^{1,-2}$.

Two-cut subgraphs are decomposed in the same manner. New upper-cut, under-cut and two-cut subgraphs are revealed during decomposition.

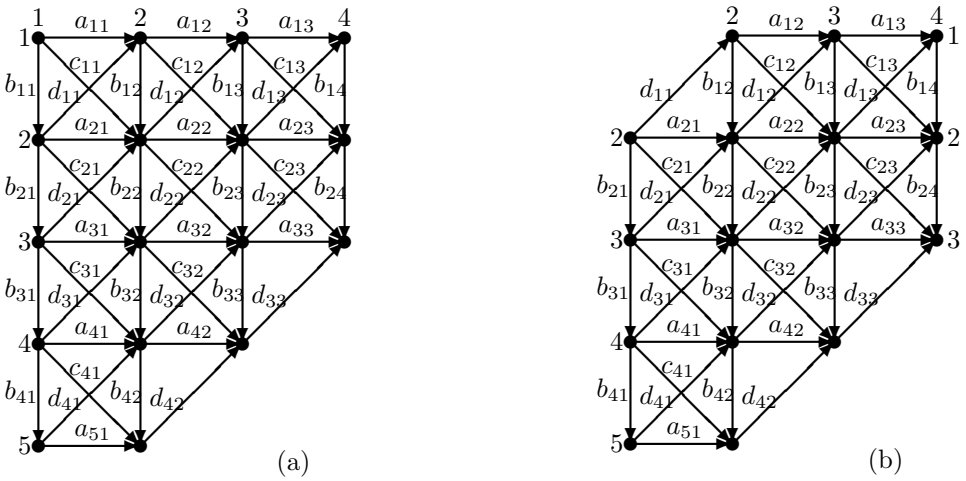


Fig. 5 Examples of newly formed graphs: (a) an under-cut king graph, (b) a two-cut king graph.

Thus, in the general case (when the size of a revealed subgraph is not less than its depth m, which is a constant) each subgraph is decomposed into $2m$ subgraphs with the same depth m. An expression of each decomposed subgraph of size n consists of $6m-4$ subexpressions related to revealed subgraphs ($m-2$ triads of pairs of subexpressions and four additional pairs corresponding to upper and lower subgraphs) of size approximately $n/2$ and $3m-2$ additional labels corresponding to connecting edges. As in Section 4, the expression can be simplified by factoring out subexpressions which appear more than once (three times or twice in our case). Depending on where the longer repeating expression is located, factoring out is done on the left or on the right, so that the longer common subexpression is taken outside the brackets. In the full-depth general case, the number of recursive subexpressions is thereby reduced by

$2m-2$ and is exactly $4m-2$; boundary cases or smaller-depth recursive calls produce no more than $4m-2$ recursive subexpressions.

The closure of these subproblem types under one recursive decomposition is summarized in Table 5. The entries are schematic: boundary cases may produce fewer children or lower depth, but never more than the stated number.

Table 5 Closure of king-subgraph types under one geometric decomposition.

Parent type	possible left child types	possible right child types	max. recursive children
rectangular	rectangular, under-cut	rectangular, upper-cut	$4m-2$
upper-cut	upper-cut, two-cut	rectangular, upper-cut	$\leq 4m-2$
under-cut	rectangular, under-cut	under-cut, two-cut	$\leq 4m-2$
two-cut	under-cut, two-cut	upper-cut, two-cut	$\leq 4m-2$

The count $4m-2$ is attained only in the full-depth rectangular/general case after the repeated left or right factors have been removed. Upper-cut, under-cut, two-cut, boundary, and smaller-depth calls may have fewer recursive children. This is why the proof below uses the uniform maximum $M_m(n)$ rather than an equality recurrence for each individual subproblem type.

Theorem 19 *The length of the expression $Ex(K_{m,n})$ (m is considered as a constant) derived by the decomposition method is $O_m\left(n^{\log_2(4m-2)}\right)$.*

Proof For fixed m, let $\mathcal{K}_m(n)$ be the finite family of all king-type subgraphs of horizontal size at most n and depth at most m that can occur in the recursive procedure: rectangular king graphs, upper-cut king graphs, under-cut king graphs, and two-cut king graphs, with arbitrary admissible cut parameters. Define

$$M_m(n) = \max\{\text{len}(Ex(H)) \mid H \in \mathcal{K}_m(n)\},$$

where the expression is the one produced by the decomposition method. Since m is fixed, all subgraphs of horizontal size at most m have expression length bounded by a constant C_m.

Consider now a recursive call on a subgraph $H \in \mathcal{K}_m(n)$ whose horizontal size is larger than m. The vertical decomposition described above produces, after the same left or right factoring, at most $4m-2$ recursive subexpressions. By Table 5, each such subexpression belongs again to $\mathcal{K}_m(\lceil n/2 \rceil)$: it is rectangular, upper-cut, under-cut, or two-cut, and its horizontal size is at most $\lceil n/2 \rceil$. The number of connecting labels is at most $3m-2$, and the labels created by the constant-size boundary cases are absorbed into C_m. Hence

$$M_m(n) \leq (4m-2)M_m\left(\left\lceil \frac{n}{2} \right\rceil\right) + C_m.$$

By the master theorem,

$$M_m(n) = O_m\left(n^{\log_2(4m-2)}\right).$$

Since $K_{m,n} \in \mathcal{K}_m(n)$, the same bound holds for $Ex(K_{m,n})$. □

We denote by $F((p_1,p_2),(q_1,q_2))$, $F^h((p_1,p_2),(q_1,q_2))$, $F^{-h}((p_1,p_2),(q_1,q_2))$, and $F^{h_1,-h_2}((p_1,p_2),(q_1,q_2))$ subexpressions related to king subgraphs, upper-cut subgraphs, under-cut subgraphs, and two-cut subgraphs, respectively, with a source (p_1,p_2) and a target (q_1,q_2). The superscripts have the same meaning as in designations of subgraphs themselves.

For example, in the graph shown in Figure 3(b), one of the local three-way contributions before factoring has the form

$$\begin{aligned} &F^{-2}((1,1),(2,3))a_{23}F^1((2,4),(5,6)) + \\ &F^{-2}((1,1),(2,3))c_{23}F^2((3,4),(5,6)) + \\ &F^{-2}((1,1),(2,3))d_{13}F((1,4),(5,6)). \end{aligned} \tag{12}$$

After factoring the common left subexpression, (12) becomes

$$F^{-2}((1,1),(2,3))\left(a_{23}F^1((2,4),(5,6)) + c_{23}F^2((3,4),(5,6)) + d_{13}F((1,4),(5,6))\right).$$

This illustrates the reduction of repeated recursive occurrences which is used in the general decomposition count.

Below we present the decomposition procedure for generating an expression for a king graph of depth 2.

1. Initial subexpressions of horizontal size 1:

$$\begin{aligned} F((p_1,p_2),(p_1+1,p_2)) &\leftarrow b_{p_1,p_2}, \\ F^1((p_1,p_2),(p_1,p_2)) &\leftarrow 1, \\ F^{-1}((p_1,p_2),(p_1,p_2)) &\leftarrow 1, \\ F^{1,-1}((p_1,p_2),(p_1,p_2)) &\leftarrow 1. \end{aligned}$$

2. Initial subexpressions of horizontal size 2 for cut subgraphs:

$$\begin{aligned} F^1((p_1,p_2),(p_1,p_2+1)) &\leftarrow a_{p_1,p_2} + d_{p_1-1,p_2}b_{p_1-1,p_2+1}, \\ F^{-1}((p_1,p_2),(p_1,p_2+1)) &\leftarrow a_{p_1,p_2} + b_{p_1,p_2}d_{p_1,p_2}, \\ F^{1,-1}((p_1,p_2),(p_1-1,p_2+1)) &\leftarrow d_{p_1-1,p_2}. \end{aligned}$$

3. If $q_2 > p_2$, set for compactness

$$\begin{gathered} S=(p_1,p_2), \quad T=(q_1,q_2), \quad R_0=(p_1,j), \quad R_1=(p_1+1,j), \\ U_0=(p_1,j+1), \quad U_1=(p_1+1,j+1). \end{gathered}$$

Then

$$\begin{aligned} F(S,T) \leftarrow\ & F^{-1}(S,R_0)\big(a_{p_1,j}F(U_0,T) + c_{p_1,j}F^1(U_1,T)\big) \\ & + F(S,R_1)\big(a_{p_1+1,j}F^1(U_1,T) + d_{p_1,j}F(U_0,T)\big). \end{aligned}$$

4. If $q_2 > p_2 + 2$, with the same abbreviations,

$$F^1(S,T) \leftarrow \big(F^{1,-1}(S,R_0)a_{p_1,j} + F^1(S,R_1)d_{p_1,j}\big)F(U_0,T) + \big(F^{1,-1}(S,R_0)c_{p_1,j} + F^1(S,R_1)a_{p_1+1,j}\big)F^1(U_1,T).$$

5. If $q_2 > p_2 + 2$, with the same abbreviations,

$$F^{-1}(S,T) \leftarrow F^{-1}(S,R_0)\big(a_{p_1,j}F^{-1}(U_0,T) + c_{p_1,j}F^{1,-1}(U_1,T)\big) + F(S,R_1)\big(a_{p_1+1,j}F^{1,-1}(U_1,T) + d_{p_1,j}F^{-1}(U_0,T)\big).$$

6. If $q_2 > p_2 + 2$, with the same abbreviations,

$$F^{1,-1}(S,T) \leftarrow F^{1,-1}(S,R_0)\big(a_{p_1,j}F^{-1}(U_0,T) + c_{p_1,j}F^{1,-1}(U_1,T)\big) + F^1(S,R_1)\big(a_{p_1+1,j}F^{1,-1}(U_1,T) + d_{p_1,j}F^{-1}(U_0,T)\big).$$

The few boundary cases of horizontal size 3 are evaluated directly by the same path-sum definition. They contribute only fixed constants to the recurrences below, so we do not display their expanded forms here; writing them out would obscure the recursive structure without affecting the length analysis.

While Steps 1, 2 describe the initial subexpressions, the general cases are processed in Steps 3 - 6. King and two-cut king graphs have symmetric boundary structure; consequently, factoring repeated subexpressions on the left or on the right gives expressions of the same length for those types. For upper-cut king graphs the right-hand extraction is the compact one, while for under-cut king graphs the corresponding compact extraction is on the left. Thus, in Steps 3, 5, 6 factoring is done on the left with $j = \left\lceil \frac{q_2+p_2-1}{2} \right\rceil$, and in Step 4 it is done on the right with $j = \left\lfloor \frac{q_2+p_2-1}{2} \right\rfloor$.

Upper-cut and under-cut subgraphs with corresponding parameters have the same decomposition length: reversing the horizontal direction, reversing each word, and relabeling edges pairs the two recursive procedures without changing the number of label occurrences. Hence one notation is enough for the two paired cut types. Denote by $L_m(n)$, $L_m^h(n)$, and $L_m^{h_1,h_2}(n)$ the lengths of the expressions $Ex(K_{m,n})$, $Ex(K_{m,n}^h)$ or $Ex(K_{m,n}^{-h})$, and $Ex(K_{m,n}^{h_1,-h_2})$ derived by the decomposition method.

As follows from the depth-two decomposition procedure, for n that is a power of two ($n = 2^p$ for some positive integer $p \geq 2$)

$$\begin{cases} L_2(n) = 3L_2\left(\frac{n}{2}\right) + 3L_2^1\left(\frac{n}{2}\right) + 4 \\ L_2^1(n) = L_2\left(\frac{n}{2}\right) + 3L_2^1\left(\frac{n}{2}\right) + 2L_2^{1,1}\left(\frac{n}{2}\right) + 4 \\ L_2^{1,1}(n) = 3L_2^1\left(\frac{n}{2}\right) + 3L_2^{1,1}\left(\frac{n}{2}\right) + 4, \end{cases} \tag{13}$$

where $L_2(2) = 7$, $L_2^1(2) = 3$, $L_2^{1,1}(2) = 1$.

The following explicit formulas ($n = 2^p$, $p \geq 1$) for (13) are obtained by the method for solving systems of simultaneous linear recurrences [38]:

$$L_2(n) = \frac{19}{30}n^{\log_2 6} + \frac{4}{3}n^{\log_2 3} - \frac{4}{5},$$

$$L_2^1(n) = \frac{19}{30}n^{\log_2 6} - \frac{4}{5},$$
$$L_2^{1,1}(n) = \frac{19}{30}n^{\log_2 6} - \frac{2}{3}n^{\log_2 3} - \frac{4}{5}.$$

For example, $Ex(K_{2,4})$ derived by the decomposition method is

$$(a_{11} + b_{11}d_{11})(a_{12}(a_{13}b_{14} + c_{13} + b_{13}(a_{23} + d_{13}b_{14})) + c_{12}(a_{23} + d_{13}b_{14})) + (a_{11}b_{12} + c_{11} + b_{11}(a_{21} + d_{11}b_{12}))(a_{22}(a_{23} + d_{13}b_{14}) + d_{12}(a_{13}b_{14} + c_{13} + b_{13}(a_{23} + d_{13}b_{14}))).$$

It contains 34 labels. For comparison, $Ex(K_{2,4})$ derived by the backtracking method contains 79 labels.

Thus, the length of $Ex(K_{m,n})$ derived by the decomposition method grows polynomially with n for every fixed m. The exponent is $\log_2(4m-2)$, so it grows only logarithmically with the fixed depth parameter m.

7 Column-Transfer Formulas and a Depth Lower Bound for King Graphs

The preceding geometric theorem is retained because it follows the same explicit separator-and-factorization philosophy as the TGG decomposition theorem and because it explains the recursive king-subgraphs that arise in the original method. The column-transfer theorem below gives the best king-graph upper bound in the paper: it uses the fixed-depth structure more directly and replaces geometric separators by the standard divide-and-conquer simulation of a fixed-width algebraic branching program by formulas; compare the general ABP viewpoint in [4, 49].

For $1 \le r \le y \le m$ and $1 \le j \le n$, put

$$B_{r,y}^{(j)} = \prod_{q=r}^{y-1} b_{q,j},$$

where an empty product is 1. Thus $B_{r,y}^{(j)}$ is the vertical path product from row r to row y inside column j. For $1 \le j < n$, define an $m \times m$ transfer matrix M_j by

$$M_j(r,z) = \sum_{y=r}^{m} B_{r,y}^{(j)} \left(\delta_{z,y}a_{y,j} + \delta_{z,y+1}c_{y,j} + \delta_{z,y-1}d_{y-1,j}\right),$$

where a term with an invalid edge index is omitted and δ is the usual Kronecker symbol. Finally, let

$$v_t(r) = B_{r,m}^{(n)}.$$

Every s-t path in $K_{m,n}$ can be uniquely decomposed as follows: in column j it first moves vertically downward, possibly by an empty vertical path, then takes exactly

one edge from column j to column $j+1$; after the last horizontal/diagonal crossing, it follows a final vertical path in column n to the target. Consequently,

$$P_{K_{m,n}} = e_1^T M_1 M_2 \cdots M_{n-1} v_t, \tag{14}$$

where e_1 is the first standard basis vector.

Theorem 20 (column-transfer upper bound) *For every fixed $m \geq 1$, the canonical path polynomial of the directed king graph $K_{m,n}$ has an expression of length*

$$O_m\left(n^{1+\log_2 m}\right).$$

For $m > 1$, this improves the exponent in Theorem 19, since $1 + \log_2 m < \log_2(4m-2)$.

Proof For fixed m, every entry of every transfer matrix M_j has length bounded by a constant depending only on m. Let $S_m(N)$ be the maximum length needed to compute one entry of a product of N consecutive transfer matrices. For $N = 1$, $S_m(N) = O_m(1)$. For $N > 1$, split the product in the middle. If $Q = M_p \cdots M_q$ and h is a middle index, then

$$Q(r,z) = \sum_{u=1}^{m} (M_p \cdots M_h)(r,u)(M_{h+1} \cdots M_q)(u,z).$$

Thus

$$S_m(N) \leq 2mS_m\left(\left\lceil \frac{N}{2} \right\rceil\right) + O_m(1).$$

The master theorem gives

$$S_m(N) = O_m\left(N^{\log_2(2m)}\right) = O_m\left(N^{1+\log_2 m}\right).$$

Multiplication by the final vector v_t and summation over the last row state add only a factor depending on m. With $N = n-1$, identity (14) gives the stated bound. □

The preceding upper bound is tight for depth 2 and has a nontrivial unrestricted lower-bound counterpart in every larger fixed depth. For $N \geq 1$ and $\varepsilon \in \{0,1\}$, let

$$\mathcal{P}_N^{\varepsilon} = \{w \in \{0,1\}^N : |w|_1 \equiv \varepsilon \pmod 2\}.$$

Lemma 21 (regular-expression lower bound for parity) For each $\varepsilon \in \{0,1\}$,

$$\mathrm{aw}(\mathcal{P}_N^{\varepsilon}) = \Omega(N^2).$$

Proof For homogeneous binary languages, Ellul, Krawetz, Shallit and Wang describe a position-wise translation from a regular expression of length s into a Boolean formula of size $O(s)$ for the characteristic function of the language [11, Sec. 3]. Applied to $\mathcal{P}_N^{\varepsilon}$, this yields a Boolean formula for the N-variable XOR/parity function. Khrapchenko's lower bound for XOR gives Boolean formula size $\Omega(N^2)$ [25]. Therefore $s = \Omega(N^2)$. The same reduction and its use for the XOR language are also summarized in the regular-expression lower-bound survey of Cseresnyes and Seiwert [6, Sec. 1.1]; related lower bounds by communication-complexity methods appear in [19]. □

Theorem 22 (unrestricted king-graph lower bound) *For every fixed $m \geq 2$,*

$$\ell(P_{K_{m,n}}) = \Omega(n^2).$$

Consequently,

$$\ell(P_{K_{2,n}}) = \Theta(n^2).$$

Proof It is enough to prove the lower bound for a restriction of $P_{K_{m,n}}$. For $m > 2$, delete all rows below row 2, except for the final vertical path from $(2, n)$ to (m, n), whose labels are set to 1. Also delete the final edge from $(1, n)$ to $(2, n)$ by setting its label to 0. This reduces the source-target condition to the following depth-two condition: after the $n - 1$ column-to-column moves, the path must be in row 2.

Now set all remaining vertical labels $b_{1,j}$ to 0. The only surviving column-to-column transitions in rows 1 and 2 are the horizontal edges $a_{1,j}, a_{2,j}$, which keep the row, and the diagonal edges $c_{1,j}, d_{1,j}$, which switch between the two rows. Map $a_{1,j}$ and $a_{2,j}$ to the alphabet symbol 0, and map $c_{1,j}$ and $d_{1,j}$ to the alphabet symbol 1.

Under this substitution, a surviving path is encoded by a binary word of length $n - 1$. Since the path starts in row 1, it ends in row 2 if and only if the number of row switches is odd. Hence the image language is exactly $\mathcal{P}^1_{n-1}$.

Let F be any expression for $P_{K_{m,n}}$. Applying the above restriction and deleting zero summands and unit factors turns F into a regular expression for $\mathcal{P}^1_{n-1}$ with alphabetic width at most $|F|$. Therefore

$$|F| \geq \mathrm{aw}(\mathcal{P}^1_{n-1}) = \Omega((n-1)^2) = \Omega(n^2).$$

Taking the minimum over all F proves the lower bound. For $m = 2$, Theorem 20 gives the matching upper bound $O(n^2)$, so $\ell(P_{K_{2,n}}) = \Theta(n^2)$. □

For depths $m > 2$, Theorem 22 is a full formula-size lower bound, but it does not match the column-transfer upper bound $O_m(n^{1+\log_2 m})$. Closing this remaining gap would require sharper formula-complexity lower bounds for fixed-width noncommutative path polynomials. The next result gives a complementary depth-restricted lower bound. We use the standard polynomial IMM_d, the iterated product of d symbolic 2×2 matrices. Chillara, Limaye and Srinivasan proved that, for every $\Delta \leq \log d$, every product-depth Δ multilinear formula for IMM_d has size $\exp(\Omega(\Delta d^{1/\Delta}))$ [8]. Here product-depth means the maximum number of multiplication gates on a root-to-leaf path.

Lemma 23 (monotone multilinearization) Let F be a monotone formula over $\mathbb{N}\langle X\rangle$ computing a multilinear polynomial. Then every product gate of F is syntactically multilinear: the two factors of the gate use disjoint sets of variables. After commutativizing all products, the resulting formula has the same length and product-depth and computes the commutative image of the polynomial.

Proof If a product gate had a variable occurring in both factors, then expansion of the formula would contain a positive monomial with that variable repeated. Since the semiring is monotone and has no cancellation, this monomial could not disappear from the computed polynomial, contradicting multilinearity. Commutativization only changes the

order of variables in monomials and does not change the formula tree, its length, or its product-depth. □

Theorem 24 (depth-restricted lower bound) *For every fixed $m \geq 2$ and every $1 \leq \Delta \leq \log(n-1)$, every product-depth Δ monotone noncommutative formula over the free semiring computing $P_{K_{m,n}}$, measured by label occurrences, has length*

$$\exp\left(\Omega\left(\Delta(n-1)^{1/\Delta}\right)\right).$$

In particular, polynomial-length monotone formulas for $P_{K_{m,n}}$ require product-depth $\Omega(\log n)$.

Proof By Theorem 3, it is enough to prove the claim for a restriction of $P_{K_{m,n}}$. If $m > 2$, delete all rows below row 2 except for the forced final vertical path from $(2, n)$ to (m, n), whose labels are set to 1. This reduces the problem to depth 2. In $K_{2,n}$, set all vertical labels $b_{1,j}$ to 0. The remaining column-to-column transitions are described by the symbolic matrices

$$A_j = \begin{pmatrix} a_{1,j} & c_{1,j} \\ d_{1,j} & a_{2,j} \end{pmatrix}, \qquad 1 \leq j < n,$$

and the restricted path polynomial is the $(1, 2)$ entry of $A_1 A_2 \cdots A_{n-1}$. After a relabeling of the entries of the last matrix, this is a standard entry of IMM_{n-1}.

The restricted polynomial is multilinear. By Lemma 23, every monotone noncommutative formula computing it commutativizes to a multilinear formula of the same length and product-depth for the usual commutative 2×2 iterated matrix multiplication polynomial. The lower bound of [8] therefore applies with $d = n - 1$. □

8 Applications of Grid-Like Graphs: Shortest Paths, Scheduling, and Reliability

Grid-like graphs, which model systems with a structured, often rectangular, layout, are applied in diverse areas, including routing [26], coloring [22], circuit design [24], chemistry [42], text analysis [55], and others. Their regular structure makes them suitable for representing spatial or logical relationships and for applying efficient algorithms for tasks such as shortest-path finding, data processing, and structured robustness analysis. Reliability is discussed separately below through two-terminal cuts and zero substitutions, not as part of this generic list of grid-like graph applications.

Specifically, we aim to use grid-structured graphs to model a task-scheduling system, namely, a multi-priority system of robotic lines in which priorities change dynamically. Each vertex of an $m \times n$ graph represents a machine, and each directed edge represents a feasible transfer link for moving a manufactured product between machines. In this context, edge labels serve as weights, each representing a generalized cost computed as a complex function of various parameters.

Row i $(i = 1, ..., m)$ corresponds to a line of priority i. A product may be transferred from machine j to machine $j + 1$ on the same line or, if necessary, to any lower line. Grid graphs allow a transition only to the machine of the same number in the new line, followed by a transfer to the next machine on that line. By contrast, TGGs and king graphs permit a direct passage to the next machine in the new line. Any transition

to a lower line decreases the task's priority, since it reduces the number of lines that remain available for future transitions. In systems modeled by grid graphs and TGGs, returning to a higher line is impossible. In systems represented by king graphs, priority can be increased, but only by one level per transition.

In the systems described above, shortest-path problems arise naturally. These problems can be solved symbolically by evaluating the st-dag expression over the min-plus semiring: formal addition is replaced by min, and formal multiplication is replaced by ordinary addition. Equivalently, one may work in the max-plus convention after negating all weights [1]. The symbolic expression is then a precompiled dynamic program for the workflow st-dag. If edge weights change but the graph structure does not, the expression can be re-evaluated without rebuilding the recursive decomposition. Thus, a compact st-dag expression provides an efficient shortest-path computation scheme on the workflow st-dag.

In [37], we employed this symbolic technique to schedule a single robotic line modeled by a Fibonacci graph. We plan to extend this work to the models schematically described above, which are based on grid-like graphs.

The same symbolic representation is also useful for failure and robustness questions. Setting an edge label to 0 in the formal path polynomial deletes all paths using that edge, while in a min-plus shortest-path interpretation a failed or forbidden edge can be represented by an infinite cost. By Theorem 4, inclusion-wise minimal sets of failures that disconnect the target from the source are exactly the minimal zero-substitution sets that annihilate the path polynomial.

This gives a reliability interpretation of the algebraic machinery. In a probabilistic network, min-cuts are minimal simultaneous edge-failure patterns that force system failure. Earlier work on Fibonacci and generalized Fibonacci graphs computed such min-cut families and reliability-related quantities [30, 32, 33, 36]; the recent fractal-grid abstract extends this viewpoint to recursive grid families of varying range [40]. The present paper contributes the complementary path-side compression: instead of enumerating all minimal blockers, it constructs compact formal expressions for all feasible paths and then uses cut substitutions to recover the minimal-blocker interpretation.

9 Conclusion and Future Work

Recursive methods for generating algebraic expressions of triangulated grid graphs and king graphs have been presented. Both types of graphs have m rows and n columns, where m is treated as a constant. The backtracking method produces expressions of length $O\left(n^m\right)$ for TGGs and expressions of at least exponential length in n for king graphs. The geometric decomposition method provides expressions of quasi-linear length, $O_m\left(n \log^{m-1} n\right)$, for TGGs and expressions of length $O_m\left(n^{\log_2(4m-2)}\right)$ for king graphs. For TGGs of depths $m \leq 4$, this quasi-linear bound is globally optimal: every expression has length $\Omega_m\left(n \log^{m-1} n\right)$. For every fixed depth, the same quasi-linear bound is optimal inside the balanced column-interval decomposition model. The unrestricted conclusion for all fixed depths would follow from the corresponding all-fixed-k lower bound for binomial languages.

For directed king graphs, the column-transfer construction improves the upper bound to $O_m\left(n^{1+\log_2 m}\right)$ by viewing fixed-depth king graphs as fixed-width algebraic branching programs. On the lower-bound side, every expression must contain at least all edge labels, restrictions preserve lower bounds, parity-language restrictions give the unrestricted formula lower bound $\ell(P_{K_{m,n}}) = \Omega(n^2)$ for all $m \geq 2$, and therefore $\ell(P_{K_{2,n}}) = \Theta(n^2)$. King-graph polynomials also inherit depth-restricted formula lower bounds from 2×2 iterated matrix multiplication.

Both methods rely on recursive algorithms. In an explicit-output implementation, the algorithms are output-sensitive: the time needed to write the formula is linear in the number of produced label occurrences, up to standard bookkeeping for parentheses and pointers. If repeated subexpressions are stored as a shared syntax DAG, the storage measure becomes a circuit/DAG measure rather than the formula length studied in this paper. Thus the stated bounds should be read as formula-length and explicit-output bounds; shared-DAG compression is a separate implementation question.

The fractal-grid min-cut viewpoint provides another natural continuation and also strengthens the reliability narrative. The earlier Korenblit–Levit work on Fibonacci min-cuts, reliability polynomials, nested graphs, and generalized Fibonacci graphs [30, 32–36] shows that min-cut growth is not merely a side observation: it is a structural quantity that directly affects cut-based reliability computations. The 47ACC abstract [40] shows that, for generalized Fibonacci, grid, and trapezoidal grid graphs parameterized by range, the number of inclusion-wise minimal source-target cuts changes from linear to quadratic to exponential growth as the range passes from 1 to 2 and then beyond 2. The cut-substitution theorem above makes this a statement about minimal ways to annihilate path polynomials. A natural next problem is to compare these cut-growth thresholds with expression-length thresholds: does exponential minimal-cut growth force new formula-size lower bounds, or can recursive factorization still compress the path polynomial effectively in fixed-depth or fixed-range regimes?

Future work will focus on the two remaining unrestricted optimality questions isolated by these bounds. The first is the all-depth global TGG question: the balanced interval-decomposition theorem proved here is sharp for the recursive decomposition model, while the unrestricted formula-size statement would follow by proving the corresponding fixed-k binomial-language regular-expression lower bound. The second is the remaining king-graph gap for $m > 2$, where the present unrestricted lower bound is $\Omega(n^2)$ and the best upper bound proved here is $O_m(n^{1+\log_2 m})$. Further compactification of the decomposition method may also be possible by avoiding explicit connecting-edge separators and identifying larger adjoining subgraphs, possibly of more complicated structure, that can be factored as units.

Acknowledgements.

The authors thank the anonymous reviewers of the earlier conference version for their careful reading and useful comments, which helped motivate several clarifications included in this journal version.

References

[1] Akian, M., Bapat, R., Gaubert, S.: Max-Plus Algebra. In: Hogben, L. et al. (eds.) Handbook of Linear Algebra. Discrete Mathematics and Its Applications, vol. 39, chapter 25, Chapman & Hall/CRC, Boca Raton (2007)

[2] Anderson, D.F., Levy, R., Shapiro, J.: Zero-Divisor Graphs, von Neumann Regular Rings, and Boolean Algebras. J. Pure Appl. Algebra **180**, 221-241 (2003)

[3] Bein, W.W., Kamburowski, J., Stallmann, M.F.M.: Optimal Reduction of Two-Terminal Directed Acyclic Graphs. SIAM J. Comput. **21**(6), 1112–1129 (1992)

[4] Bringmann, K., Ikenmeyer, C., Zuiddam, J.: On Algebraic Branching Programs of Small Width. J. ACM **65**(5), Article 32, 1–29 (2018). doi:10.1145/3209663

[5] Celerier, C., Joyner, D., Melles, C., Phillips, D., Walsh, S.: Edge-weighted Cayley graphs and p-ary bent functions. Integers **16**, Paper A20, 1–56 (2016)

[6] Cseresnyes, E., Seiwert, H.: Regular Expression Length via Arithmetic Formula Complexity. J. Comput. Syst. Sci. **125**, 1–24 (2022). doi:10.1016/j.jcss.2021.10.004

[7] Burns, R.N., Steiner, G.: Single Machine Scheduling with Series-Parallel Precedence Constraints. Oper. Res. **29**(6), 1195–1207 (1981)

[8] Chillara, S., Limaye, N., Srinivasan, S.: Small-Depth Multilinear Formula Lower Bounds for Iterated Matrix Multiplication with Applications. SIAM J. Comput. **48**(1), 70–92 (2019). doi:10.1137/18M1191567

[9] Dubrova, E., Macchiarulo, L.: A comment on graph-based algorithm for boolean function manipulation. IEEE Trans. Comput. **49**(11), 1290–1292 (2000)

[10] Duffin, R.J.: Topology of Series-Parallel Networks. J. Math. Anal. Appl. **10**, 303–318 (1965)

[11] Ellul, K., Krawetz, B., Shallit, J., Wang, M.-W.: Regular Expressions: New Results and Open Problems. J. Autom. Lang. Comb. **9**(2–3), 233–256 (2004). doi:10.25596/jalc-2004-233

[12] Fernau, H.: Algorithms for Learning Regular Expressions from Positive Data. Information and Computation **207**, 521–541 (2009)

[13] Fernau, H., Ryan, J.F., Sugeng, K.A.: A Sum Labelling for the Generalised Friendship Graph. Discrete Math. **308**, 734–740 (2008)

[14] Finta, L., Liu, Z., Milis, I., Bampis, E.: Scheduling UET-UCT Series-Parallel Graphs on Two Processors. Theoret. Comput. Sci. **162**(2), 323–340 (1996)

[15] Forbes, M.A., Shpilka, A., Tzameret, I., Wigderson, A.: Proof Complexity Lower Bounds from Algebraic Circuit Complexity. Theory of Computing **17**(10), 1–88 (2021)

[16] Golumbic, M.C., Gurvich, V.: Read-Once Functions. In: Crama, Y., Hammer, P.L. (eds.) Boolean Functions: Theory, Algorithms and Applications, pp. 519–560. Cambridge University Press, New York (2011)

[17] Golumbic, M.C., Mintz, A., Rotics, U.: Factoring and Recognition of Read-Once Functions Using Cographs and Normality and the Readability of Functions Associated with Partial k-Trees. Discrete Appl. Math. **154**(10), 1465–1477 (2006)

[18] Golumbic, M.C., Perl, Y.: Generalized Fibonacci Maximum Path Graphs. Discrete Math. **28**, 237–245 (1979)

[19] Gruber, H., Johannsen, J.: Optimal Lower Bounds on Regular Expression Size Using Communication Complexity. In: Amadio, R. (ed.) FoSSaCS 2008. LNCS, vol. 4962, pp. 273–286. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78499-9_20

[20] Gulan, S.: Series Parallel Digraphs with Loops. Graphs Encoded by Regular Expression. Theory Comput. Syst. **53**, 126–158 (2013)

[21] Hermelin, D., Levy, A., Weimann, O., Yuster, R.: Distance Oracles for Vertex-Labeled Graphs. In: Aceto, L., Henzinger, M., Sgall, J. (eds.) ICALP 2011, Part II. LNCS, vol. 6756, pp. 490–501. Springer, Heidelberg (2011). doi:10.1007/978-3-642-22012-8_39

[22] Herva, P., Karl, J.: On Forced Periodicity of Perfect Colorings. Theory Comput. Syst. **67**, 732–759 (2023)

[23] Holzer, M., Jakobi, S.: Grid Graphs with Diagonal Edges and the Complexity of Xmas Mazes. In: Kranakis, E., Krizanc, D., Luccio, F. (eds.) FUN 2012. LNCS, vol. 7288, pp. 223–234. Springer, Heidelberg (2012). doi:10.1007/978-3-642-30347-0_23

[24] Kaufmann, M., Gao, S., Thulasiraman, K.: On Steiner Minimal Trees in Grid Graphs and its Application to VLSI Routing. In: Ding-Zhu Du, Xiang-Sun Zhang (eds.) ISAAC 1994. LNCS, vol. 834, pp. 351–359. Springer, Heidelberg (1994)

[25] Khrapchenko, V.M.: Method of Determining Lower Bounds for the Complexity of P-schemes. Math. Notes Acad. Sci. USSR **10**, 474–479 (1971). doi:10.1007/BF01747074

[26] Kayaturan, G.C., Vernitski, A.: Encoding Paths with Binary Arrays in a King's Graph for Error-Free Data Transmission. Annals of Mathematics and Artificial Intelligence **94**(1), 111–125 (2025). doi:10.1007/s10472-025-09985-7

[27] Korenblit, M.: Decomposition Methods for Generating Algebraic Expressions of Full Square Rhomboids and Other Graphs. Discrete Appl. Math. **228**, 60–72 (2017)

[28] Korenblit, M.: Generating Algebraic Expressions for Labeled Grid Graphs. In: Kim, D., Zelikovski, A. (eds.) COCOA 2018. LNCS, vol. 11346, pp. 138–153. Springer, Heidelberg (2018)

[29] Korenblit, M.: Algebraic Expressions of Grid Graphs with Diagonal Edges. In: Misra, N., Pandey, A. (eds.) Algorithms and Discrete Applied Mathematics, CALDAM 2026. LNCS, vol. 16445, pp. 278–292. Springer, Cham (2026). doi:10.1007/978-3-032-17156-6_21

[30] Korenblit, M., Levit, V.E.: The st-Connectedness Problem for a Fibonacci Graph. WSEAS Transactions on Mathematics **1**(2), 89–93 (2002)

[31] Korenblit, M., Levit, V.E.: On Algebraic Expressions of Series-Parallel and Fibonacci Graphs. In: Calude, C.S., Dinneen, M.J., Vajnovzki, V. (eds.) DMTCS 2003. LNCS, vol. 2731, pp. 215–224. Springer, Heidelberg (2003)

[32] Korenblit, M., Levit, V.E.: On Sizes of Mincuts in a Fibonacci Graph. International Journal of Computer Research **12**(3), 151–158 (2003)

[33] Korenblit, M., Levit, V.E.: On the Reliability Polynomial of Fibonacci and Some Other Graphs. WSEAS Transactions on Mathematics **2**(1), 43–47 (2003)

[34] Korenblit, M., Levit, V.E.: Minimal Cuts in Two-Terminal Directed Acyclic Graphs. Congressus Numerantium **193**, 67–87 (2008)

[35] Korenblit, M., Levit, V.E.: A Note on the Recognition of Nested Graphs. In: Lipshteyn, M., Levit, V.E., McConnell, R.M. (eds.) Graph Theory, Computational Intelligence and Thought. LNCS, vol. 5420, pp. 36–40. Springer, Heidelberg (2009). doi:10.1007/978-3-642-02029-2_4

[36] Korenblit, M., Levit, V.E.: Mincuts in Generalized Fibonacci Graphs of Degree 3. Journal of Computational Methods in Sciences and Engineering **11**(5–6), 271–280 (2011). doi:10.3233/JCM-2011-0396

[37] Korenblit, M., Levit, V.E.: Symbolic Solutions of Shortest-Path Problems and Their Applications. In: Tuba, M., Akashe, S., Joshi, A. (eds.) ICT System and Sustainability, Proc. ICT4SD 2019, Vol. 1, Advances in Intelligent Systems and Computing, vol. 1077, pp. 299–307. Springer, Heidelberg (2019)

[38] Korenblit, M., Levit, V.E.: Some Systems of Simultaneous Linear Recurrences and Their Applications to Computing of Graph Expression Lengths. JCMCC **114**, 193–223 (2020)

[39] Korenblit, M., Levit, V.E.: On Lengths of Edge-Labeled Graph Expressions. Discrete Appl. Math. **319**, 583–594 (2022). doi:10.1016/j.dam.2022.04.015

[40] Korenblit, M., Levit, V.E.: On the Number of Min-Cuts in Fractal Grid Graphs. Abstract, 47th Australasian Combinatorics Conference (47ACC), Victoria University of Wellington–Te Herenga Waka, Wellington, New Zealand, 1–5 December 2025, Abstracts Booklet, p. 41. https://sms.wgtn.ac.nz/Events/ACC47/

[41] Law, H.F., Leung, S.L., Ostrovskii, M.I.: Spanning Tree Congestion of Planar Graphs. Involve J. Math. **7**(2), 205–226 (2014)

[42] Liu, S., Ou, J., Lin, Y.: On K-Resonance of Grid Graphs on the Plane, Torus and Cylinder. J. Math. Chem. **52**(7), 1807–1816 (2014)

[43] Mertens, S.: Domination Polynomials of the Grid, the Cylinder, the Torus, and the King Graph. J. Integer Sequences **27**, Article 24.7.6 (2024)

[44] Mousavi, H.: Lower Bounds on Regular Expression Size. CoRR abs/1712.00811 (2017). arXiv:1712.00811

[45] Mintz, A., Golumbic, M.C.: Factoring Boolean Functions Using Graph Partitioning. Discrete Appl. Math. **149**(1–3), 131–153 (2005). doi:10.1016/j.dam.2005.02.007

[46] Mundici, D.: Functions Computed by Monotone Boolean Formulas with no Repeated Variables. Theoret. Comput. Sci. **66**, 113–114 (1989)

[47] Mundici, D.: Solution of Rota's Problem on the Order of Series-Parallel Networks. Adv. Appl. Math. **12**, 455–463 (1991)

[48] Naumann, V.: Measuring the Distance to Series-Parallelity by Path Expressions. In: Mayr, E.W., Schmidt, G., Tinhofer, G. (eds.) WG 1994. LNCS, vol. 903, pp. 269–281. Springer, Heidelberg (1995)

[49] Nisan, N.: Lower Bounds for Non-Commutative Computation. In: Proc. 23rd Annual ACM Symposium on Theory of Computing (STOC 1991), pp. 410–418. ACM, New York (1991). doi:10.1145/103418.103462

[50] Oikawa, M.K., Ferreira, J.E., Malkowski, S., Pu, C.: Towards Algorithmic Generation of Business Processes: From Business Step Dependencies to Process Algebra Expressions. In: Dayal, U., Eder, J., Koehler, J., Reijers, H.A. (eds.) BPM 2009. LNCS, vol. 5701, pp. 80–96. Springer, Heidelberg (2009)

[51] Provan, J.S., Ball, M.O.: Computing Network Reliability in Time Polynomial in the Number of Cuts. Operations Research **32**(3), 516–526 (1984). doi:10.1287/opre.32.3.516

[52] Rajan, M.A., Lokesha, V., Niranjan, K.M.: On Study of Vertex Labeling of Graph Operations. J. Sci. Res **3**(2), 291–301 (2011)

[53] Satyanarayana, A., Wood, R.K.: A Linear Time Algorithm for Computing K-Terminal Reliability in Series-Parallel Networks. SIAM J. Comput. **14** (4), 818–832 (1985)

[54] Savicky, P., Woods, A.R.: The Number of Boolean Functions Computed by Formulas of a Given Size. Random Struct. Algorithms **13**, 349–382 (1998)

[55] Schmidt, J.P.: All Highest Scoring Paths in Weighted Grid Graphs and Their Application to Finding All Approximate Repeats in Strings. SIAM J. Comput. **27**(4), 972–992 (1998)

[56] Surapati, R., Akiri, S., Velampudi, S.R., Nagarjuna, V.B.V., Swetha, P.S.S., Kapu, B.: Enhancing Reliability of Series-Parallel Systems: A Novel Mathematical Model for Redundancy Allocation. Eur. J. Pure Appl. Math. **18**(1), Article 5672 (2025). doi:10.29020/nybg.ejpam.v18i1.5672

[57] Wang, J.B., Wang, J.J.: Single-machine scheduling with precedence constraints and position-dependent processing times. Appl. Math. Model. **37**, 649–658 (2013)

[58] Weisstein, E.W.: Grid Graph. From MathWorld – A Wolfram Web Resource. http://mathworld.wolfram.com/GridGraph.html

[59] Weisstein, E.W.: Triangular Grid. From MathWorld – A Wolfram Web Resource. https://mathworld.wolfram.com/TriangularGrid.html

[60] Weisstein, E.W.: King Graph. From MathWorld – A Wolfram Web Resource. https://mathworld.wolfram.com/KingGraph.html

[61] Zhou, Y., Zhang, Z., Lin, T.R., Ma, L.: Maintenance optimisation of a multi-state series–parallel system considering economic dependence and state-dependent inspection intervals. Reliab. Eng. Syst. Saf. **111**, 248–259 (2013). doi:10.1016/j.ress.2012.10.006