

A polynomial algorithm for general d -position in trees

Daniel A. Pyle

Independent Researcher

ORCID: 0009-0005-6456-1370

21 October 2026

Abstract

A vertex set is in general d -position if no three of its vertices lie on a shortest path of length at most d . We give an exact algorithm for an n -vertex tree in $O(n(D+2)^4)$ time and $O(n(D+2)^2)$ space, where $D = \min\{d, n-1\}$. In particular, the problem is polynomial-time solvable when the threshold is part of the input. This answers the tree-complexity question posed by Klavžar, Rall, and Yero in 2021 and retained in the August 2026 version of the general position survey. The dynamic program uses two truncated distances at each rooted boundary: the nearest selected vertex and the nearest far endpoint of an ancestor-comparable selected pair. These distances characterize every forbidden triple created by joining child subtrees. The algorithm reconstructs an optimal set and extends to signed vertex weights and exact counting of optimal sets. It also computes the full cardinality generating polynomial in $O(n^2(D+2)^4)$ coefficient operations. Exhaustive comparisons with direct subset enumeration support the implementation independently of the proof.

Keywords. General position; general d -position; tree; dynamic programming; exact counting.

MSC 2020. 05C12, 05C85, 68R10.

AI provenance and review status. Daniel A. Pyle initiated and directed this project. Lyra, an OpenAI-powered AI assistant, carried out the technical literature research, proof development, computation, and manuscript drafting. Separate AI-assisted audits examined the arguments and computations. These checks are neither human peer review nor proof-assistant formalization, and no independent human verification of the mathematics is claimed. Lyra is an AI contributor, not a human author. This manuscript has not been peer reviewed.

1 Introduction

All graphs considered here are finite, simple, and unweighted unless vertex weights are explicitly introduced. For a graph G and a nonnegative integer d , a set $S \subseteq V(G)$ is in *general d -position* if no three distinct vertices of S lie on a geodesic of length at most d . Write $\text{gp}_d(G)$ for its largest possible cardinality. The terminology and the distance-bounded problem were introduced by Klavžar, Rall, and Yero [2]. We include $d = 0, 1$ as harmless boundary cases: every set is feasible for these thresholds.

For a tree T , the path between any two vertices is unique. Thus S is feasible precisely when

$$x, y, z \in S \text{ distinct, } y \in V(P_T(x, z)) \implies \text{dist}_T(x, z) > d. \quad (1)$$

The restriction is on the length between the two extreme vertices, not on either individual gap between successive selected vertices.

Klavžar, Rall, and Yero ask whether $\text{gp}_d(T)$ can be computed in polynomial time for trees when $2 < d < \text{diam}(T)$ [2, Section 6.1]. The question is still recorded in Section 3.1 of the August 2026 survey by Chandran, Klavžar, and Tuite [3]. We resolve the tree part by an explicit dynamic program with a polynomial dependence on both the order and the threshold.

Theorem 1.1. *Let T be a tree on $n \geq 1$ vertices, let $d \geq 0$ be an integer, and set $D = \min\{d, n-1\}$. A maximum general d -position set of T can be found, with a witnessing set, in*

$$O(n(D+2)^4) \text{ time and } O(n(D+2)^2) \text{ space.}$$

The time bound uses a word RAM with $O(\log(n+1))$ -bit words after reading and capping the threshold. In particular, the bound is $O(n^5)$ uniformly in d .

The same state space supports maximum signed vertex weight, the number of optimum sets, and the number of all feasible sets. For integer or rational weights, the operation bound is unchanged, while the bit cost depends on the encoding of the weights. For counting, the same arithmetic-operation bound applies but the integers can require $O(n)$ bits. Section 5 gives these distinctions explicitly, as well as a bound for the full cardinality generating polynomial.

The endpoints provide useful context. At $d = 2$, feasibility on a tree says exactly that the selected induced subgraph has maximum degree at most one. This is the dissociation-set problem; the equality with general 2-position on triangle-free graphs is recorded in [2, Proposition 5.2 and Corollary 5.3]. When $d \geq \text{diam}(T)$, the condition is ordinary general position, whose value for a nontrivial tree is its number of leaves [1, Corollary 3.7]. These endpoints do not settle intermediate thresholds.

For each fixed d , feasibility can also be expressed by a fixed logical formula forbidding selected triples on paths of bounded length. Standard logical optimization methods on trees therefore give an alternative fixed- d tractability argument; see the fixed-formula framework of [4]. Such an argument does not by itself give a polynomial bound in the variable d . The contribution here is a uniform, elementary algorithm with a two-coordinate boundary description. We make no claim that its running time is optimal. The priority assessment uses freely readable primary sources and a literature search through 21 October 2026; it cannot exclude unindexed or unpublished concurrent work.

2 Two distances at a rooted boundary

Root T at an arbitrary vertex r_0 . For a vertex r , a *partial rooted subtree* is the subgraph consisting of r and zero or more complete child subtrees of r . Such a piece is obtained while children are processed one at a time. Its boundary is r . Requiring whole child subtrees, rather than an arbitrary vertex subset, is essential to the separation argument below.

A vertex x is a *strict ancestor* of y if $x \neq y$ and x lies on the path from r_0 to y . Given a set S in a partial rooted subtree with boundary r , define

$$\alpha_r(S) = \min\{\text{dist}(r, x) : x \in S\}, \tag{2}$$

$$\beta_r(S) = \min\{\text{dist}(r, y) : x, y \in S, x \text{ is a strict ancestor of } y\}. \tag{3}$$

A minimum over an empty set is $+\infty$. The second coordinate records the depth of the *far endpoint* of a selected comparable pair. It is not the distance between that pair, and it does not record pairs from different branches.

Every path in T has length at most $n - 1$, so replacing d by $D = \min\{d, n - 1\}$ preserves feasibility. Put $L = D + 1$ and define

$$\langle t \rangle = \min\{t, L\} \quad (t \geq 0), \quad \langle +\infty \rangle = L.$$

We use the clipped signature

$$\sigma_r(S) = (a, b) = (\langle \alpha_r(S) \rangle, \langle \beta_r(S) \rangle) \in \{0, 1, \dots, L\}^2. \quad (4)$$

The sentinel L represents both a missing witness and a distance larger than D . These cases need not be distinguished. In particular, $a = 0$ if and only if $r \in S$, while $b = 0$ is impossible for an attainable signature.

Lemma 2.1. *For nonnegative extended distances s, t ,*

$$s + t \leq D \iff \langle s \rangle + \langle t \rangle \leq D.$$

Clipping also commutes with taking a minimum and with a one-edge shift followed by clipping.

Proof. If either distance is larger than D , its clipped value is $D + 1$, so both inequalities fail. Otherwise clipping leaves both distances unchanged. The minimum identity follows directly from monotonicity, and $\langle t + 1 \rangle = \langle \langle t \rangle + 1 \rangle$, including when $t = +\infty$. $\square$

Example 2.2. A nearest-vertex distance alone is insufficient. Let $D = 5$ and consider one branch from r with selected vertices at depths 2 and 3, or instead at depths 2 and 5. Both choices have $a = 2$ and are internally feasible. A selected vertex in a different branch at depth 2 creates a forbidden triple with the first choice, because $3 + 2 = 5$, but not with the second, because $5 + 2 > 5$. The second coordinate distinguishes exactly these cases.

For a partial rooted subtree P with boundary r , let

$$F_P(a, b) = \max\{\text{wt}(S) : S \subseteq V(P), S \text{ is feasible}, \sigma_r(S) = (a, b)\}, \quad (5)$$

where $\text{wt}(S) = \sum_{v \in S} w(v)$. For the unweighted problem, $w(v) = 1$. A state with no realizing set is absent, or equivalently has value $-\infty$. Feasibility inside a partial rooted subtree is the same whether paths are measured in that piece or in T , since the piece contains the unique path between any two of its vertices.

3 The merge lemma

Let A be a partial rooted subtree with boundary r , and let B be a child subtree not yet included in A . Their vertex sets are disjoint: r belongs to A , not to B . Let $S_A \subseteq V(A)$ and $S_B \subseteq V(B)$ be feasible. For this section, both signatures are measured from r , so write

$$\sigma_r(S_A) = (a, b), \quad \sigma_r(S_B) = (c, e).$$

The second signature is obtained by adding one to each coordinate of the signature at the child root and clipping. Ancestor comparability within B does not change when its reference boundary moves to r .

Lemma 3.1 (Compatibility and signature update). *The union $S_A \cup S_B$ is in general D -position if and only if*

$$b + c > D \quad \text{and} \quad a + e > D. \quad (6)$$

For a compatible pair, the union has signature

$$\sigma_r(S_A \cup S_B) = (\min\{a, c\}, \min\{b, e, h(a, c)\}), \quad h(a, c) = \begin{cases} c, & a = 0, \\ L, & a \neq 0. \end{cases} \quad (7)$$

Proof. A forbidden triple contained entirely in A or in B is excluded by the hypotheses. Every other triple has two vertices on one side and one on the other.

First consider two selected vertices $x, y \in A$ and a selected vertex $z \in B$. A path from B into A passes through r . The three vertices are collinear if and only if x and y are ancestor-comparable from r : after interchanging their names if necessary, x is a strict ancestor of y . In that case the path from y to z contains x and has length

$$\text{dist}(r, y) + \text{dist}(r, z).$$

Consequently a forbidden triple of this type exists if and only if $\beta_r(S_A) + \alpha_r(S_B) \leq D$. The two minima can be realized independently because they concern disjoint pieces; if a minimum is infinite, no such triple exists. Lemma 2.1 makes this condition equivalent to $b + c \leq D$.

The case with one selected vertex in A and two in B is symmetric and gives $a + e \leq D$. These cases include all triples containing r , because r lies in A . This proves (6).

The nearest selected vertex of the union is the nearer of the nearest vertices on its two sides, giving the first coordinate of (7). A comparable selected pair either lies wholly in A , lies wholly in B , or uses both pieces. Vertices in different child subtrees of r are incomparable. Therefore the third possibility occurs precisely when $r \in S_A$ and S_B is nonempty. Its smallest far-endpoint depth is $\alpha_r(S_B)$. The condition $r \in S_A$ is exactly $a = 0$; if the second piece is empty, $c = L$ already contributes nothing to the minimum. Taking the clipped minimum yields the second coordinate. $\square$

The two inequalities are necessary as well as sufficient. In particular, counting only triples that pass through a selected boundary vertex would miss a comparable selected pair farther down one branch together with a vertex in another branch. The term $b + c$ detects these triples without recording their individual vertices.

4 Dynamic program and proof of the main theorem

At a piece consisting only of its boundary r , initialize

$$F_{\{r\}}(L, L) = 0, \quad F_{\{r\}}(0, L) = w(r). \quad (8)$$

These entries represent the empty set and the singleton $\{r\}$. For each child u of r , first compute the table for the whole subtree at u . A child state (s, t) becomes

$$(c, e) = (\min\{L, s + 1\}, \min\{L, t + 1\}) \quad (9)$$

when measured from r . For each current state (a, b) , discard the pair if either inequality in (6) fails. Otherwise update the state in (7) with the candidate value $F_A(a, b) + F_{T_u}(s, t)$. When multiple candidates reach one state, retain the maximum.

Algorithm 1. General d -position on a tree.*Input:* a rooted tree T , threshold $d \geq 0$, and weights $w(v)$ (all one for cardinality).*Output:* the maximum feasible weight and, if requested, a witnessing set.

1. Set $D \leftarrow \min\{d, n - 1\}$ and $L \leftarrow D + 1$.
2. Process vertices r in postorder. Start a table F with $(L, L) \mapsto 0$ and $(0, L) \mapsto w(r)$.
3. For each child u of r , in a fixed order:
 - (a) Start an empty table H .
 - (b) For every present (a, b) in F and (s, t) in F_u , set $c \leftarrow \min\{L, s + 1\}$ and $e \leftarrow \min\{L, t + 1\}$.
 - (c) If $b + c > D$ and $a + e > D$, set $a' \leftarrow \min\{a, c\}$ and $b' \leftarrow \min\{b, e, h(a, c)\}$, and update

$$H(a', b') \leftarrow \max\{H(a', b'), F(a, b) + F_u(s, t)\}.$$
4. After all children of r are processed, store $F_r \leftarrow F$.
5. At the overall root r_0 , return $\max_{a,b} F_{r_0}(a, b)$ and backtrack one maximizing state if requested.

Treat an absent value as $-\infty$. Store a maximizing predecessor pair if a witness is required.

- (d) Replace F by H .

The pseudocode iterates over unshifted child states and computes their shifts within the loop. Distinct child states can collapse to the same shifted signature. This causes no difficulty for optimization: every candidate is considered. It is also convenient for the counting variants, where their contributions must be added rather than discarded.

Theorem 4.1 (Table invariant). *After every child merge, each present entry satisfies (5), and every attainable clipped signature is present.*

Proof. Induct on completed child subtrees, and within a vertex on its processed children. Initialization (8) lists exactly the two subsets of the one-vertex piece.

Assume that the two input tables satisfy the invariant. A feasible set S in the enlarged piece has a unique decomposition $S_A = S \cap V(A)$ and $S_B = S \cap V(B)$. Both restrictions are feasible. Their input signatures occur by induction, their pair passes Lemma 3.1, and the algorithm computes the correct signature of S .

Conversely, for every pair of present states that passes the compatibility tests, any realizing sets on the two sides have a feasible union with the computed signature. Moreover, replacing either realizing set by an optimum representative of the same input signature changes neither compatibility nor the output signature. The value of an optimum union for that state pair is therefore the sum of its two input optima. Maximizing over all pairs gives exactly the optimum for every output state. It neither omits a feasible signature nor introduces an unattainable one, completing the induction. $\square$

Proof of Theorem 1.1. At the overall root there is no remaining outside constraint. By Theorem 4.1, the maximum over its states is the required optimum.

There are at most $(D + 2)^2$ signatures in a table and at most $(D + 2)^4$ input state pairs per merge. There are exactly $n - 1$ child merges. Each pair requires only a constant number of comparisons, additions, and table accesses. Initialization is lower order. Cardinalities, coordinates, and table indices have $O(\log(n + 1))$ bits, giving the stated word-RAM time bound.

Retain the table at each initialization and after each child merge. There are $n + (n - 1) = 2n - 1$ such stages, so the total storage is $O(n(D+2)^2)$ entries. For each output entry, retain one maximizing predecessor pair and the identities of its two input stages. Select a maximizing final state and follow these pointers recursively. The selected decomposition visits each merge once and each one-vertex initialization once. The initialization choice determines whether to include that vertex. Thus backtracking takes $O(n)$ pointer visits and produces an optimal set without increasing the asymptotic storage.

Finally $D \leq n - 1$, so the time bound is $O(n^5)$. Reading a binary-encoded threshold contributes its input length, but even an extremely large threshold is then capped before any table is allocated. $\square$

The empty tree, if admitted, has optimum zero and can be handled separately. Root choice and child order affect intermediate tables but not the optimum. An implementation may omit unreachable signatures and release tables not needed for backtracking; the proof does not require either optimization.

5 Weights and exact counting

The merge structure concerns only feasibility. It can therefore be reused with several kinds of table values, provided the aggregation respects the unique decomposition into the two pieces.

5.1 Signed vertex weights and arithmetic cost

Corollary 5.1. *For arbitrary signed integer vertex weights, a maximum-weight general d -position set can be found with the same $O(n(D+2)^4)$ arithmetic-operation and $O(n(D+2)^2)$ entry bounds. Rational weights are also allowed. The empty set remains an available choice.*

Proof. Theorem 4.1 uses only additivity of the objective and comparison of candidate sums. It never assumes that the weights are nonnegative. $\square$

If each integer weight has at most B bits including its sign, every partial sum has $O(B + \log(n+1))$ bits. A direct bit implementation of score additions and comparisons therefore takes $O(n(D+2)^4(B + \log(n+1)))$ bit operations, apart from standard indexing costs of the same or smaller order. For rational weights, multiplying all weights by the product of their positive denominators reduces to integer weights. The resulting bit lengths are polynomial in the total input length. This reduction is sufficient for a polynomial bit-complexity claim; no unit-cost claim for arbitrary-size rational arithmetic is intended.

5.2 Optimal sets and all feasible sets

For each state, store a pair (v, m) consisting of the optimum score and the number of sets attaining it. Both initialization counts equal one. An accepted pair of entries (v_1, m_1) and (v_2, m_2) contributes

$$(v_1 + v_2, m_1 m_2).$$

A larger score replaces the stored score and count; an equal score adds its count; a smaller score contributes nothing to the optimum count. At the root, sum counts only over states attaining the global optimum.

Corollary 5.2. *The number of maximum-cardinality or maximum-weight general d -position sets is computable in $O(n(D+2)^4)$ arithmetic operations. The total number of feasible sets is computable within the same arithmetic-operation bound.*

Proof. For a fixed compatible state pair, restriction to the disjoint pieces is a bijection between realizing unions and ordered pairs of realizing subsets. Thus counts multiply within a candidate and add between distinct candidates. An optimum union with a fixed pair of input signatures must optimize both restrictions separately. Each set has a unique restriction and a unique input signature on each side, so ties and collapsed shifted signatures do not cause double counting. To count all feasible sets, simply replace the optimum-score entries by total counts, multiply across compatible pairs, and add all candidates. At the root, sum over all signatures. $\square$

Every count is at most 2^n and has $O(n)$ bits. Hence the displayed bound counts arithmetic operations, not constant-time machine-word operations. If $M(n)$ bounds multiplication of two $O(n)$ -bit integers, a conservative bit bound for unweighted optimization with optimum counting, or for counting all feasible sets, is $O(n(D+2)^4 M(n))$, with the coordinate and score costs absorbed. Signed weights contribute their score-arithmetic cost in addition.

5.3 The cardinality generating polynomial

Define

$$P_{T,d}(z) = \sum_{\substack{S \subseteq V(T) \\ S \text{ in general } d\text{-position}}} z^{|S|}. \quad (10)$$

This polynomial records all cardinalities, not just the optimum. Its degree is $\text{gp}_d(T)$, its constant coefficient is one, and $P_{T,d}(1)$ counts all feasible sets.

Theorem 5.3. *The polynomial $P_{T,d}(z)$ can be computed in $O(n^2(D+2)^4)$ coefficient arithmetic operations, using $O(n^2(D+2)^2)$ stored coefficients if all stages are retained.*

Proof. Associate to each signature the sum of $z^{|S|}$ over all realizing feasible sets. Initialize the empty-set state with 1 and the singleton state with z . For each compatible state pair, multiply its polynomials and add the product to the output signature. The unique-decomposition argument in Corollary 5.2 proves this recurrence coefficient by coefficient. Sum all root-state polynomials to obtain (10).

For the operation bound, consider a merge whose two disjoint pieces have p and q vertices. Ordinary coefficient convolution for one state pair uses $O((p+1)(q+1))$ operations. The child-merging process forms a binary decomposition tree with the original vertices as its leaves. Every unordered pair of vertices is separated between the two inputs at exactly one merge, namely their first common merge. Consequently

$$\sum_{\text{merges}} pq = \binom{n}{2}.$$

Since $p, q \geq 1$, we have $(p+1)(q+1) \leq 4pq$. Multiplying by the maximum $(D+2)^4$ state pairs and summing over merges gives the claimed coefficient-operation bound. There are $2n-1$ retained stages, at most $(D+2)^2$ states per stage, and at most $n+1$ coefficients per state, giving the storage bound. $\square$

Each coefficient has $O(n)$ bits, so coefficient operations are not unit-cost bit operations. A direct multiplication-based bit bound is $O(n^2(D+2)^4M(n))$. The theorem concerns the computation of this polynomial on trees; it is not a claim to have introduced general-position generating polynomials.

6 Examples and reproducible verification

The proof establishes correctness for all trees and all thresholds. Computation serves a separate purpose: testing the implementation against a definition-level oracle and checking boundary cases that are easy to misencode.

6.1 Elementary examples

For $d = 0$ or 1 , every subset is feasible and $P_{T,d}(z) = (1+z)^n$. In a path, writing selected vertices in path order as $x_1, \dots, x_k$, feasibility is equivalent to

$$\text{dist}(x_i, x_{i+2}) > d \quad (1 \leq i \leq k-2). \quad (11)$$

Indeed, any three consecutive selected vertices violating this condition form a forbidden triple. Conversely, a forbidden triple has an extreme-vertex interval containing at least three selected vertices, and therefore contains three consecutive selected vertices of no larger span.

For the six-vertex path, direct enumeration and the algorithm give the following values. The count in the final row is the number of maximum-cardinality feasible subsets, not all feasible subsets.

Threshold d	0	1	2	3	4	5
$\text{gp}_d(P_6)$	6	6	4	4	3	2
Number of maximum sets	1	1	6	1	4	15

Table 1: A small instance showing the effect of the distance threshold. Values are unchanged for $d \geq 5$.

On a star $K_{1,m}$ with $m \geq 2$ and $d \geq 2$, all leaves form a feasible set. A selected center can coexist with at most one selected leaf, since two selected leaves and the center form a forbidden length-two path. Thus the optimum is $\max\{m, 2\}$. The merge recurrence exhibits the same mechanism: selecting the center sets $a = 0$, the first leaf creates a pair with $b = 1$, and a second leaf has $c = 1$, causing rejection when $d \geq 2$.

6.2 Definition-level oracles

The reproducibility package contains a standard-library Python implementation and a separately written audit. Both enumerate nonisomorphic trees by leaf insertion and deduplicate using a canonical parenthesis representation, minimized over all root choices. Every tree of order $n > 1$ has a leaf whose deletion gives a tree of order $n - 1$, so this construction reaches every isomorphism type. Canonicalization removes repeated representations.

The direct oracles do not use the two-distance merge lemma to determine feasibility. For each triple, they compute its three pairwise tree distances and sort them as $p \leq q \leq s$. The vertices are

collinear exactly when $p + q = s$; their extreme-vertex span is then s . Enumerating subsets and recording their smallest such span gives the feasible sets for every threshold.

The first implementation checks all 987 nonisomorphic trees of orders one through twelve, with order counts

$$1, 1, 1, 2, 3, 6, 11, 23, 47, 106, 235, 551.$$

For each tree it compares optimum values at every $d = 0, \dots, n$, giving 11,993 comparisons. An extension run compares the full cardinality generating polynomial for the same 11,993 tree-threshold pairs through order twelve. There are also 851 additional root-and-threshold comparisons through order seven, and 350 seeded signed-weight, optimum-count, and witness checks on random trees through order fourteen. All comparisons pass. The latter witness checks use explicit bit masks for validation; the space-efficient reconstruction in Theorem 1.1 instead uses stage backpointers.

A separate audit compares complete state tables, not merely final optima. For each attainable clipped signature it compares the optimum score, the number of sets attaining that score, and the total number of feasible sets; it also detects missing or spurious signatures. Its objectives are cardinality and deterministic seeded signed integer weights from -3 through 3 . Every vertex is used as root, and thresholds include $0, \dots, n - 1$, $2n + 3$, and 10^{40} , the latter two testing the cap. The audit passes 107,956 full-table instances across all 436 nonisomorphic trees through order eleven. A second independent run compares every coefficient of every state polynomial across all 95 nonisomorphic trees through order nine, with every root and the same threshold family, giving 7,653 complete polynomial-table comparisons. Table 2 summarizes the distinct suites; overlapping tree-threshold instances are not claimed to be distinct inputs. These computational checks support the implementation but are not a substitute for the invariant proof.

Comparison suite	Maximum order	Instances
Reference optimum versus subsets	12	11,993
Reference full cardinality polynomial	12	11,993
Additional root checks	7	851
Seeded weight, count, and witness checks	14	350
Independent complete score-and-count tables	11	107,956
Independent complete polynomial tables	9	7,653

Table 2: Completed exact checks. An instance of a table suite compares the entire table, not one entry. Random checks use seed 20261021.

7 Scope and further questions

The two-coordinate signature is sufficient because a new forbidden triple crosses a one-vertex boundary as a comparable pair on one side and a single selected vertex on the other. No history of individual selected vertices is needed once the two relevant depths are known. This gives a uniform polynomial algorithm for the tree question, rather than only a tractability result with a fixed distance threshold.

The present bound is deliberately direct: it examines all pairs of possible signatures at each merge. A faster implementation could exploit restrictions on attainable states or the order structure of the two compatibility inequalities. No improved bound is claimed here. The same proof does not automatically extend to block graphs, arbitrary graphs, weighted edge lengths, or versions

forbidding more than three collinear selected vertices. In particular, the block-graph part of the question in [2, Section 6.1] is not resolved by this manuscript.

The counting variants show that optimization is not the only information retained by the boundary description. Exact optimum counts and the complete cardinality distribution are accessible without enlarging the signature. Further work may investigate whether comparably small signatures exist for related distance-bounded general-position problems, including the broader framework of Cody and Moore [5].

Data and code availability

A companion reproducibility archive contains the reference implementation, independent audit, recorded test outputs, and instructions for rerunning them with Python 3. The compact source archive contains this standalone L^AT_EX source and requires no figure files or external bibliography database. No public repository or external submission is claimed in this version.

License

The manuscript and its original source are made available under the Creative Commons Attribution 4.0 International license, <https://creativecommons.org/licenses/by/4.0/>. Third-party works cited below retain their own licenses. The companion archive contains a separate notice about its original code.

References

- [1] P. Manuel and S. Klavžar, A general position problem in graph theory, *Bulletin of the Australian Mathematical Society* **98** (2018), 177–187. doi:10.1017/S0004972718000473. Open text: <https://www.cambridge.org/core/services/aop-cambridge-core/content/view/S0004972718000473>.
- [2] S. Klavžar, D. F. Rall, and I. G. Yero, General d -position sets, *Ars Mathematica Contemporanea* **21**(1) (2021), #P1.03. doi:10.26493/1855-3974.2384.77d. Open published text: <https://users.fmf.uni-lj.si/klavzar/preprints/2384-11213.pdf>.
- [3] U. Chandran S. V., S. Klavžar, and J. Tuite, *The General Position Problem: A Survey*, arXiv:2501.19385v5 (16 August 2026). <https://arxiv.org/abs/2501.19385v5>.
- [4] B. Courcelle, J. A. Makowsky, and U. Rotics, Linear time solvable optimization problems on graphs of bounded clique-width, *Theory of Computing Systems* **33** (2000), 125–150. doi:10.1007/s002249910009. Open text: <https://www.fceia.unr.edu.ar/~daniel/OC/Apuntes/COURCELLE.pdf>.
- [5] B. Cody and G. Moore, The k -general d -position problem for graphs, *Discrete Applied Mathematics* **366** (2025), 135–151. doi:10.1016/j.dam.2025.01.025. Open version: <https://arxiv.org/abs/2409.05644>.