

Counting and Enumerating Unique Neighborhood Networks

Md. Rifat Rahman Md. Saidur Rahman

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology, Dhaka, Bangladesh

Abstract

A Unique Neighborhood Network (UNN) is a graph in which no two vertices have the same neighborhood, so that a vertex is determined by its neighbors alone. Read gave a closed form for the number of its labeled members, and the sequence is catalogued in the OEIS. A count, however, does not produce instances, and no method is known for generating a specific UNN on demand. We treat constructive enumeration as the primitive operation. We give a recurrence for a_n , the number of labeled UNNs on n vertices, whose summands are the cardinalities of disjoint and constructively described families, and we turn it into an unranking function: an integer $r \in [1, a_n]$ is converted into an n -vertex UNN in $O(n^3)$ time. Ranks are independent, so the whole class is generated by splitting $\{1, \dots, a_n\}$ into intervals.

Keywords: unique neighborhood networks, point-determining graphs, graph enumeration, unranking, fault localization, open-locating-dominating sets.

1 Introduction

A *Unique Neighborhood Network* (UNN) is a graph in which every vertex has a distinct set of neighbors, so that each vertex can be identified with its neighborhood. The condition has appeared under several names. Sumner [5] called such graphs point determining, Bull and Pease [6] found them in mating type systems and called them M-graphs, and Rass [3] named them Unique Neighborhood Networks. This paper is about counting them and, more to the point, generating them.

The property matters wherever a vertex has to be identified by its neighbors rather than by a name of its own, and fault localization is the clearest instance. Model a facility or a multiprocessor network as a graph $G = (V, E)$ whose vertices are locations and whose edges join locations a device can sense across, and put a device at every vertex. A device reports an event at a location adjacent to it, but not at its own location, because an event at a location disables the device there. This is the open neighborhood model introduced by Seo and Slater [1], as opposed to identifying codes [2], in which a device also senses its own vertex. An event at u therefore makes exactly the devices at the locations adjacent to u report, and the operator sees that set and nothing else. The system pinpoints the event if the set determines u , and it does so for every possible event if and only if no two locations have the same neighborhood. The layout must be a UNN, and a designer who plans a facility has to produce such a layout before the devices are installed.

The class has been counted. Read [7] obtained a closed form for the number of labeled M-graphs by a generating function argument, later placed in a systematic framework by Gessel and Li [8], and the counting sequence is entry A006024 of the OEIS [9]. The question of occurrence has been settled as well: Rass [3] shows that an Erdős–Rényi graph with constant edge probability and a random geometric graph are almost surely UNNs as $n \rightarrow \infty$, while a scale-free graph fails to be one with probability above 11.49%. No prior work has been done on deterministically producing a UNN. Although a formula for counting exists, that formula is not suitable for the enumeration purpose.

Problem 1.1 (Constructive enumeration). Given n , produce an n -vertex UNN deterministically from a short input, so that the same input always yields the same graph, every admissible graph comes from some input, and no two inputs yield the same graph. That is, exhibit a bijection between a set of inputs and the set of n -vertex UNNs.

A solution to Problem 1.1 is an unranking function for the class, and it reduces the description of a graph to a single integer r . Because the map is a bijection, r names one UNN and no other, every UNN has such a name, and the whole class is swept out by letting r run over $\{1, \dots, a_n\}$. In the localization setting the designer draws r once, builds the matrix, installs the devices according to it, and publishes the pair (n, r) ; an installer, an inspector or a later revision of the plan rebuilds the adjacency matrix from (n, r) and reads off any row. Our contributions are the following.

- (1) A recurrence for a_n whose summands are the cardinalities of disjoint, exhaustive and constructive families (Section 3).
- (2) An unranking algorithm that converts $r \in [1, a_n]$ into the adjacency matrix of an n -vertex UNN in $O(n^3)$ time (Section 4). Ranks partition into independent tasks, so generation parallelizes without communication.

Section 2 fixes notation, Section 3 develops the recurrence, Section 4 gives the algorithm and its analysis, and Section 5 concludes.

2 Preliminaries

All graphs are simple, undirected and unweighted. For $G = (V, E)$ with $V = \{v_1, \dots, v_n\}$ the adjacency matrix is $A = [a_{ij}]$ with $a_{ij} = 1$ if $\{v_i, v_j\} \in E$ and $a_{ij} = 0$ otherwise, which is symmetric with zero diagonal. The neighborhood of v_i is $\text{nb}(v_i) = \{v_j \in V : \{v_i, v_j\} \in E\}$, and its indicator vector is row i of A .

Definition 2.1 (Unique Neighborhood Network). A graph $G = (V, E)$ is a *Unique Neighborhood Network* if $\text{nb}(v_i) \neq \text{nb}(v_j)$ for all $v_i, v_j \in V$ with $i \neq j$. Equivalently, no two rows of A are identical, and by symmetry no two columns are identical.

We write $\mathcal{U}_n$ for the set of labeled UNNs on the vertex set $\{1, \dots, n\}$ and $a_n = |\mathcal{U}_n|$.

Since the property depends only on pairwise distinctness of adjacency rows, enumerating $\mathcal{U}_n$ is the problem of filling an $n \times n$ symmetric 0–1 matrix whose diagonal entries are zero and whose rows, equivalently columns, are pairwise distinct. Everything below works with the matrix, and this formulation drives the construction of Section 3.

3 Counting Unique Neighborhood Networks

We classify an n -vertex UNN by what remains after vertex n is deleted. Let M be the adjacency matrix of $G \in \mathcal{U}_n$ and let M' be the $(n-1) \times (n-1)$ principal submatrix obtained by deleting row and column n . The matrix M' is symmetric with zero diagonal but need not have distinct rows, and two cases arise.

The submatrix is already a UNN.

Lemma 3.1. *Let A_{n-1} be the adjacency matrix of a UNN on $n-1$ vertices. The number of ways to append a row and column, symmetrically and with zero diagonal, so that the resulting $n \times n$ matrix is a UNN, is exactly $2^{n-1} - (n-1)$.*

Proof. The new row is determined by the vector $x = (x_1, \dots, x_{n-1}) \in \{0, 1\}^{n-1}$ of adjacencies of vertex n , since the remaining entry is forced by the zero diagonal, so there are 2^{n-1} candidates. Row i of the extended matrix is $(A_{n-1}[i], x_i)$ and row n is $(x, 0)$. These coincide if and only if $x = A_{n-1}[i]$ and $x_i = 0$, and the second condition follows from the first because $A_{n-1}[i][i] = 0$. Each existing row therefore forbids exactly one candidate, the forbidden candidates are pairwise distinct because the rows of A_{n-1} are, and no other collision is possible because rows $i \neq j$ already differ within their first $n-1$ entries. $\square$

Summing over all UNNs of order $n-1$ gives the contribution $a_{n-1}(2^{n-1} - (n-1))$.

The submatrix contains duplicates.

Lemma 3.2. *If an $(n-1) \times (n-1)$ symmetric 0–1 matrix with zero diagonal extends to an $n \times n$ UNN, then it has no three identical rows, and equivalently no three identical columns.*

Proof. Suppose rows $i < j < k$ are identical. After vertex n is added, the neighborhood of each of these vertices is either the common original neighborhood, if the vertex is not adjacent to n , or that neighborhood together with $\{n\}$, if it is. Three vertices are then assigned to at most two distinct neighborhoods, so two of them coincide and Definition 2.1 fails. The argument for columns is symmetric. $\square$

By Lemma 3.2 the identical rows of M' form disjoint pairs, which we call *twin pairs*. Let p be their number and q the number of rows that occur once. *Merging* deletes one row of each twin pair together with the column of the same index, and produces a matrix of order $k = p + q$ with $2p + q = n - 1$, so that

$$p = n - 1 - k, \quad q = 2k - n + 1, \quad (1)$$

and $p, q \geq 0$ forces $\lfloor n/2 \rfloor \leq k \leq n - 1$.

Lemma 3.3 (Collapse). *The merged $k \times k$ matrix is symmetric, has zero diagonal and has pairwise distinct rows, hence it is the adjacency matrix of a UNN on k vertices.*

Proof. Let $\{b, c\}$ be a twin pair. Rows b and c of M' are equal, so by symmetry columns b and c are equal too; every row of M' therefore carries the same entry in column b as in column c , and it makes no difference which of the two is deleted. Symmetry is inherited, since deleting a row and its column preserves it. For the diagonal, the entry of M' at position (b, c) equals the entry at (c, c) , because rows b and c are equal, and that entry is 0; the surviving diagonal entries are therefore all zero. For distinctness, let i and j index two surviving rows. They are not twins of each other, so they differ somewhere, say in column t . If t survived the merge, the two rows still differ there. If t was deleted, it belonged to a twin pair $\{b, c\}$ whose other column survived, and since every row has equal entries in columns b and c , rows i and j differ in that surviving column as well. $\square$

The number of ways to select which of the $n-1$ positions form the p twin pairs is

$$\binom{n-1}{2} \binom{n-3}{2} \cdots / p! = \frac{(n-1)!}{p! q! 2^p}, \quad (2)$$

where the product has p factors and the division by $p!$ removes the order of the pairs.

Lemma 3.4. *Let M' have $p \geq 1$ twin pairs and q rows that occur once, with $k = p + q$. The number of vectors $x \in \{0, 1\}^{n-1}$ such that appending x as row and column n yields an $n \times n$ UNN is exactly 2^k .*

Proof. Take a twin pair $\{b, c\}$. Rows b and c of the extended matrix differ if and only if $x_b \neq x_c$, which leaves two admissible assignments for the pair. A row i that occurs once is already distinct from every other row within the first $n - 1$ entries, so both values of x_i are admissible. This gives $2^p \cdot 2^q = 2^k$ vectors, and it remains to check that row n differs from all others. Row n cannot equal a row b of a twin pair, since equality would force $x_b = M'[b][b] = 0$ and $x_c = M'[b][c] = M'[c][c] = 0$, contradicting $x_b \neq x_c$. Row n cannot equal a row i that occurs once either, since $M'[i][b] = M'[i][c]$ whereas $x_b \neq x_c$. $\square$

Combining Lemma 3.3, expression (2) and Lemma 3.4, the contribution of a fixed $k < n - 1$ is $(n - 1)! 2^k a_k / (2^p p! q!)$, and substituting (1) turns the coefficient into

$$\text{coeff}(n, k) = \frac{(n - 1)!}{(2k - n + 1)! (n - 1 - k)!} \cdot 2^{2k - n + 1}. \quad (3)$$

Theorem 3.5. *Let $a_1 = a_2 = 1$. For every $n \geq 3$,*

$$a_n = a_{n-1} (2^{n-1} - n + 1) + \sum_{k=\lfloor n/2 \rfloor}^{n-2} \text{coeff}(n, k) a_k. \quad (4)$$

Proof. Map $G \in \mathcal{U}_n$ to the order k of its merged submatrix, the merged UNN, the choice of twin pairs, and the vector x . Deleting vertex n determines M' ; Lemma 3.2 bounds the multiplicity of identical rows by two, so p and hence k are determined; Lemma 3.3 places the merged matrix in $\mathcal{U}_k$; and x is read off column n . The map is injective because M is recovered from these data by expansion and appending, and it is surjective onto their product by Lemma 3.4 for $k < n - 1$ and by Lemma 3.1 for $k = n - 1$. Counting the fibres gives $a_{n-1} (2^{n-1} - n + 1)$ for $k = n - 1$ and $\text{coeff}(n, k) a_k$ for each $k < n - 1$, and the range of k follows from (1). $\square$

Table 1 lists the first values, which agree with the counts of Read [7] and with entry A006024 of the OEIS [9].

Table 1: Number of labeled UNNs on n vertices.								
n	1	2	3	4	5	6	7	8
a_n	1	1	4	32	588	21,476	1,551,368	218,608,712

Definition 3.6 (Unranking). An unranking function for $\mathcal{U}_n$ is a bijection $U_n : \{1, \dots, a_n\} \rightarrow \mathcal{U}_n$ computable without materializing $\mathcal{U}_n$ [10, 11].

Read's closed form [7] gives the same values, $a_n = \sum_k s(n, k) 2^{\binom{k}{2}}$ with $s(n, k)$ the signed Stirling numbers of the first kind, but its terms alternate in sign and none of them is the size of a family of UNNs, so a rank cannot be traced to one. Every term of Theorem 3.5 is the size of an explicit family, so a rank can be compared against the terms in turn until the family holding it is found, which is what Section 4 does.

4 Enumeration algorithm

By Theorem 3.5 an n -vertex UNN is built from a smaller UNN of order k , with $\lfloor n/2 \rfloor \leq k \leq n - 1$, by expanding the $k \times k$ matrix into an $(n - 1) \times (n - 1)$ matrix through duplication of $p = n - 1 - k$ rows and the corresponding columns, and then appending a final row and column. Accordingly a rank r is decomposed into three sub-ranks: r_{sm} , the rank of the smaller UNN; r_{dig} , the binary content of the final row and column; and r_{box} , the pattern of duplication. The three are computed independently and the matrix is assembled bottom up by recursion, with base cases $n = 1$ and $n = 2$. The values $a_1, \dots, a_n$ are precomputed once by (4).

Locating the block. Ranks are laid out in increasing order of k . Algorithm 1 subtracts block sizes until the block containing r is found and returns the rank relative to it.

Algorithm 1 FINDBLOCK: locate the rank inside a block of Theorem 3.5

Require: rank r , dimension n

Ensure: relative rank r' , sub-dimension k

```

1: for  $k \leftarrow \lfloor n/2 \rfloor$  to  $n - 1$  do
2:    $size \leftarrow a_k \cdot \text{coeff}(n, k)$ 
3:   if  $r > size$  then
4:      $r \leftarrow r - size$ 
5:   else
6:     return  $r, k$ 

```

Algorithm 1 applies the single expression $\text{coeff}(n, k)$ to every block, including $k = n - 1$, where $\text{coeff}(n, n - 1) = 2^{n-1}$ exceeds the true block size $2^{n-1} - (n - 1)$ of Lemma 3.1. This is harmless: $k = n - 1$ is the last block, the preceding subtractions leave r no larger than the true size, so the test on line 3 fails there and the loop returns the correct k . Only the exact size is needed afterwards, and Algorithm 2 uses it.

Splitting the rank. In the case $k = n - 1$ each smaller UNN admits $T = 2^{n-1} - (n - 1)$ extensions by Lemma 3.1, and no duplication is present, which we signal by $r_{\text{box}} = -1$. In the case $k < n - 1$ write $P = \text{coeff}(n, k)$ for the number of variants per smaller UNN and $B = P/2^k$ for the number of duplication patterns, which by (2) equals $(n - 1)!/(2^p p! q!)$. The three sub-ranks are a mixed radix representation of r' , so the split is a bijection onto $\{1, \dots, a_k\} \times \{1, \dots, 2^k\} \times \{1, \dots, B\}$.

Algorithm 2 DECOMPOSERANK: split a rank into sub-ranks

Require: rank r , dimension n

Ensure: $r_{\text{sm}}, k, r_{\text{dig}}, r_{\text{box}}$

```

1:  $r', k \leftarrow \text{FINDBLOCK}(r, n)$ 
2: if  $k = n - 1$  then
3:    $T \leftarrow 2^{n-1} - n + 1$ 
4:    $r_{\text{sm}} \leftarrow \lceil r'/T \rceil$ ;  $r_{\text{dig}} \leftarrow r' - (r_{\text{sm}} - 1)T$ ;  $r_{\text{box}} \leftarrow -1$ 
5: else
6:    $P \leftarrow \text{coeff}(n, k)$ ;  $B \leftarrow P/2^k$ 
7:    $r_{\text{sm}} \leftarrow \lceil r'/P \rceil$ ;  $v \leftarrow r' - P(r_{\text{sm}} - 1)$ 
8:    $r_{\text{dig}} \leftarrow \lceil v/B \rceil$ ;  $r_{\text{box}} \leftarrow v - (r_{\text{dig}} - 1)B$ 
9: return  $r_{\text{sm}}, k, r_{\text{dig}}, r_{\text{box}}$ 

```

The final row and column. By Lemma 3.4 the admissible vectors are in bijection with $\{0, 1\}^k$, one bit per group, where a group is either a position that occurs once or a twin pair. Algorithm 3 reads the bits of $r_{\text{dig}} - 1$ from the least significant end, emitting the two-bit pattern (1, 0) or (0, 1) for each of the $d = n - k - 1$ twin pairs first and a single bit for each remaining group, and reverses the result. The emitted sequence has length $2d + (2k - n + 1) = n - 1$ and is ordered with the single entries first, matching the order in which Algorithm 4 returns its two lists.

The duplication pattern. The object to be unranked is a partition of $\{1, \dots, n - 1\}$ into $q = 2k - n + 1$ positions that occur once and $p = n - 1 - k$ twin pairs. Write $N(p) = (2p)!/(2^p p!)$ for the number of ways

Algorithm 3 LASTROWPATTERN: recover the final row from r_{dig}

Require: dimension n , sub-dimension k , digit rank r_{dig} **Ensure:** binary sequence x of length $n - 1$

```
1:  $d \leftarrow n - k - 1$ ;  $t \leftarrow r_{\text{dig}} - 1$ ;  $x \leftarrow \langle \rangle$ 
2: for  $i \leftarrow 1$  to  $k$  do
3:    $b \leftarrow t \bmod 2$ ;  $t \leftarrow \lfloor t/2 \rfloor$ 
4:   if  $d > 0$  then
5:     append  $(1, 0)$  to  $x$  if  $b = 0$ , else append  $(0, 1)$ ;  $d \leftarrow d - 1$ 
6:   else
7:     append  $(b)$  to  $x$ 
8: reverse  $x$ 
9: return  $x$ 
```

to pair $2p$ elements. Algorithm 4 proceeds in two phases under a canonical rule: single positions are chosen in increasing order, and within each pair the smaller element is the least position not yet placed. In the first phase the candidate position u is tested; the number of arrangements in which u is single is $\binom{n-1-i}{m-1} N(p)$, where i positions have been consumed and m single positions remain, and the rank is reduced by that count when u is not single. In the second phase the least unplaced position is paired with the element selected by dividing the residual rank by $N(j-1)$.

When $k = n - 1$ we have $p = 0$, $q = n - 1$ and $r_{\text{box}} = -1$, so the test on line 5 never succeeds, every position is placed as single, and Algorithm 4 returns $S = [1, \dots, n - 1]$, $D = []$ without a special case.

Algorithm 4 UNRANKPAIRING: recover the duplication pattern from r_{box}

Require: dimension n , sub-dimension k , box rank r_{box} **Ensure:** list S of single positions in increasing order, list D of twin pairs

```
1:  $q \leftarrow 2k - n + 1$ ;  $p \leftarrow n - k - 1$ ;  $L \leftarrow [1, \dots, n - 1]$ ;  $S \leftarrow []$ ;  $D \leftarrow []$ 
2:  $u \leftarrow 1$ ;  $m \leftarrow q$ ;  $i \leftarrow 1$ 
3: while  $m > 0$  do ▷ phase 1: single positions
4:    $c \leftarrow \binom{n-1-i}{m-1} \cdot N(p)$ 
5:   if  $r_{\text{box}} > c$  then
6:      $r_{\text{box}} \leftarrow r_{\text{box}} - c$  ▷  $u$  is not single
7:   else
8:     append  $u$  to  $S$ ; remove  $u$  from  $L$ ;  $m \leftarrow m - 1$ 
9:    $u \leftarrow u + 1$ ;  $i \leftarrow i + 1$ 
10: for  $j \leftarrow p$  down to 1 do ▷ phase 2: twin pairs
11:    $\text{idx} \leftarrow \lceil r_{\text{box}} / N(j - 1) \rceil$ 
12:    $r_{\text{box}} \leftarrow r_{\text{box}} \bmod N(j - 1)$ ; if  $r_{\text{box}} = 0$  then  $r_{\text{box}} \leftarrow N(j - 1)$ 
13:   append the pair  $(L[1], L[\text{idx} + 1])$  to  $D$ ; remove  $L[\text{idx} + 1]$  and then  $L[1]$  from  $L$ 
14: return  $S, D$ 
```

Assembly. Algorithm 5 obtains the $k \times k$ matrix by a recursive call, builds an index map that sends a single position to itself and both members of a twin pair to the smaller of the two, compresses the image to $\{1, \dots, k\}$, and expands. The final row is placed by pairing the position list S followed by D with the sequence returned by Algorithm 3 and sorting by position. In the case $k = n - 1$ the digit rank indexes only the admissible vectors, so the decoded integer is first corrected by a skip over the $n - 1$ values already present

as rows.

Algorithm 5 BUILDMATRIX: the unranking function U_n

Require: dimension n , rank $r \in \{1, \dots, a_n\}$

Ensure: $n \times n$ adjacency matrix M of a UNN

```

1: if  $n = 1$  then
2:   return  $[[0]]$ 
3: if  $n = 2$  then
4:   return  $[[0, 1], [1, 0]]$ 
5:  $r_{\text{sm}}, k, r_{\text{dig}}, r_{\text{box}} \leftarrow \text{DECOMPOSERANK}(r, n)$ 
6:  $M_k \leftarrow \text{BUILDMATRIX}(k, r_{\text{sm}})$  ▷ recursive call
7:  $S, D \leftarrow \text{UNRANKPAIRING}(n, k, r_{\text{box}})$ 
8: for each  $u \in S$  do  $\text{map}[u] \leftarrow u$ 
9: for each  $(a, b) \in D$  do  $\text{map}[a] \leftarrow a; \text{map}[b] \leftarrow a$ 
10:  $\text{reps} \leftarrow$  distinct values of  $\text{map}$  in increasing order
11: for  $u \leftarrow 1$  to  $n - 1$  do  $\text{map}[u] \leftarrow$  position of  $\text{map}[u]$  in  $\text{reps}$ 
12: for  $i, j \leftarrow 1$  to  $n - 1$  do
13:    $M_{\text{exp}}[i][j] \leftarrow M_k[\text{map}[i]][\text{map}[j]]$ 
14:  $x \leftarrow \text{LASTROWPATTERN}(n, k, r_{\text{dig}})$ 
15: if  $k = n - 1$  then ▷ skip over the forbidden vectors
16:    $v \leftarrow$  value of  $x$  read as a binary numeral
17:    $\text{used} \leftarrow$  sorted values of the rows of  $M_{\text{exp}}$  read as binary numerals
18:   for each  $e \in \text{used}$  in increasing order do if  $v \geq e$  then  $v \leftarrow v + 1$ 
19:    $x \leftarrow$  binary expansion of  $v$  on  $n - 1$  digits
20:  $\text{pos} \leftarrow S$  followed by the entries of  $D$  in the order produced
21:  $y \leftarrow$  the sequence  $x$  permuted so that  $\text{pos}$  is increasing
22: for  $i \leftarrow 1$  to  $n - 1$  do  $M[i] \leftarrow M_{\text{exp}}[i]$  with  $y[i]$  appended
23:  $M[n] \leftarrow y$  with 0 appended
24: return  $M$ 

```

Correctness.

Theorem 4.1. For every $n \geq 1$ the map U_n computed by Algorithm 5 is a bijection from $\{1, \dots, a_n\}$ onto $\mathcal{U}_n$.

Proof. Induction on n . The base cases are immediate, since $a_1 = a_2 = 1$ and the matrices listed are the only ones. Assume the claim for all orders below n . Algorithm 1 compares r against the block sizes, which by Theorem 3.5 sum to a_n and are the cardinalities of disjoint families whose union is $\mathcal{U}_n$, so it identifies exactly one k and a relative rank inside that block. Algorithm 2 is a mixed radix split of that interval, hence a bijection onto the product of the three sub-rank ranges, whose sizes are a_k , then 2^k or T , and B . The first factor is handled by the induction hypothesis, the second by Lemma 3.4 for $k < n - 1$ and by Lemma 3.1 with the skip over for $k = n - 1$, and the third by the canonical rule of Algorithm 4, whose two phases partition the range because $\binom{n-1-i}{m-1} N(p)$ counts exactly the arrangements in which the current position is single and $N(j - 1)$ counts the completions of each choice of partner. The assembly step inverts the merge of Lemma 3.3. Hence every rank yields a distinct element of $\mathcal{U}_n$ and every element is reached. $\square$

Complexity. Let $T(n)$ be the time to build a UNN of order n . Outside the recursive call one invocation performs FINDBLOCK in $O(n)$, DECOMPOSERANK in $O(1)$ beyond that, UNRANKPAIRING in $O(n^2)$ with

the candidate list kept in sorted order, LASTROWPATTERN in $O(n)$, the expansion in $O(n^2)$, the skip over in $O(n^2)$ and the final assembly in $O(n^2)$. The dominant cost per level is therefore $O(n^2)$ and

$$T(n) \leq T(k) + O(n^2), \quad \lfloor n/2 \rfloor \leq k \leq n-1. \quad (5)$$

In worst case, when $k = n-1$ the recursion loses one vertex per level, giving $T(n) = \sum_{i=2}^n O(i^2) = O(n^3)$.

Implementation. We implemented Algorithms 1–5 in Python. For $n \leq 7$ every rank in $\{1, \dots, a_n\}$ was expanded and each output checked for symmetry, zero diagonal and pairwise distinct rows; all outputs were valid and the number of distinct matrices equalled a_n in each case, which is Theorem 4.1 verified over the whole class for those orders.

5 Conclusion

Prior work established when UNN graphs occur and how many of them exist. We have addressed the complementary problem of producing one on purpose. The technical content is a recurrence for the number of labeled UNNs whose terms are the sizes of disjoint and constructive families, and an algorithm that turns this recurrence into a bijection between $\{1, \dots, a_n\}$ and the class itself.

There are some limitations in the current work. First, a uniformly drawn UNN is dense, with expected degree close to $n/2$, so it carries close to $n^2/4$ edges. An application that needs a sparse instance should not draw uniformly, and unranking within a prescribed edge count or degree bound is an open extension of the recurrence.

Second, the construction does not extend to the stronger condition that no neighborhood be contained in another [4, 3]. The extension and collapse lemmas of Section 3 do not carry over, because the number of admissible last rows is then no longer a function of n and k alone but depends on the containment order of the smaller instance, so a recurrence and an unranking function for that class remain open. An instance of the stronger class can meanwhile be obtained by drawing a rank and testing the condition, since that class sits inside the one treated here.

Third, the construction is confined to undirected graphs. Symmetry of the adjacency matrix is what makes the merge of Lemma 3.3 well defined and the argument doesn't survive when predecessor and successor sets are distinguished, so the directed case is not covered.

References

- [1] S. J. Seo, P. J. Slater, Open neighborhood locating-dominating sets, *Australasian Journal of Combinatorics* 46 (2010) 109–119.
- [2] M. G. Karpovsky, K. Chakrabarty, L. B. Levitin, On a new class of codes for identifying vertices in graphs, *IEEE Transactions on Information Theory* 44 (1998) 599–611.
- [3] S. Rass, Tell me who you are friends with and I will tell you who you are: unique neighborhoods in random graphs, *Theoretical Computer Science* 988 (2024) 114386.
- [4] S. Rass, Perfectly secure communication, based on graph-topological addressing in unique-neighborhood networks, arXiv:1810.05602, 2023.
- [5] D. P. Sumner, Point determination in graphs, *Discrete Mathematics* 5 (2) (1973) 179–187.
- [6] J. J. Bull, C. M. Pease, Combinatorics and variety of mating-type systems, *Evolution* 43 (3) (1989) 667–671.

- [7] R. C. Read, The enumeration of mating-type graphs, Research Report CORR 89-38, Department of Combinatorics and Optimization, University of Waterloo, 1989.
- [8] I. M. Gessel, J. Li, Enumeration of point-determining graphs, *Journal of Combinatorial Theory, Series A* 118 (2) (2011) 591–612.
- [9] OEIS Foundation Inc., Entry A006024, number of labeled point-determining graphs, *The On-Line Encyclopedia of Integer Sequences*, <https://oeis.org/A006024>.
- [10] A. Nijenhuis, H. S. Wilf, *Combinatorial Algorithms*, 2nd Edition, Academic Press, New York, 1978.
- [11] F. Ruskey, *Combinatorial Generation*, working version 1j-CSC 425/520, University of Victoria, 2003.