A Electronic Appendix [at reviewers' discretion]

Lemma 1. *Let $\succeq_*$ be any total order on S_n with $\mathrm{id} \succeq_* \pi$, for all $\pi \in S_n$.*

(i) Let $\tau \in S_n$ be any permutation. A family $\mathscr{F} = (\pi_1, \ldots, \pi_d)$ is k-restricted minwise independent iff $\widetilde{\mathscr{F}} = (\pi_1 \circ \tau, \ldots, \pi_d \circ \tau)$ has this property.

(ii) There exists a k-restricted minwise independent family $\mathscr{F} = (\pi_1, \ldots, \pi_d)$ iff there exists a k-restricted minwise independent family $\widetilde{\mathscr{F}} = (\widetilde{\pi_1}, \ldots, \widetilde{\pi_d})$ that fulfills $\widetilde{\pi_1} = \mathrm{id}$ and $\widetilde{\pi_i} \succeq_* \widetilde{\pi_{i+1}}$, for $i = 1, \ldots, d-1$.

Proof. (i) For an arbitrary $X \subseteq [n]$ and $x^* \in X$, after denoting $y^* := \tau(x^*)$ and $Y := \{\tau(x) : x \in X\}$, we obtain k-restricted minwise independence of $\widetilde{\mathscr{F}}$ from the fact that $|\{i \in [d] : \pi_i \circ \tau(x^*) = \min\{\pi_i \circ \tau(x) : x \in X\}\}| = |\{i \in [d] : \pi_i(y^*) = \min\{\pi_i(y) : y \in Y\}\}|$ coincides with $1/|Y|$ due to minwise independence of $\mathscr{F}$, therefore also with $1/|X| = 1/|Y|$.

(ii) Setting $\tau := \pi^{-1}$, the property holds according to (i) for the family $\widetilde{\mathscr{F}} := (\pi_1 \circ \pi_1^{-1}, \ldots, \pi_d \circ \pi_1^{-1})$ after its members indexed $2, \ldots, d$ have been rearranged into a chain with respect to $\succeq_*$.

Denote by $\mathbb{B}^{m\times n}_{\mathrm{asc}}$ (respectively $\mathbb{B}^{m\times n}_{\mathrm{desc}}$) all binary $m \times n$ matrices in which the i-th row is a lexicographic predecessor (respectively successor) of the $(i+1)$-th one, for $i \in [m-1]$.

Lemma 2. *For an arbitrary Boolean $A = (a_{ij})_{(i,j)\in[2]\times[n]} \in \{0,1\}^{2\times n}$, consider the following system of membership-constraints involving the Boolean $m_{\mathrm{asc}} \in \{0,1\}$:*

$$\begin{pmatrix} m_{\mathrm{asc}} & a_{11}\ a_{12}\ \ldots\ a_{1n} \\ 1 & a_{21}\ a_{22}\ \ldots\ a_{2n} \end{pmatrix} \in \mathbb{B}^{2\times(1+n)}_{\mathrm{asc}}, \tag{28}$$

$$\begin{pmatrix} 0 & \neg a_{11}\ \neg a_{12}\ \ldots\ \neg a_{1n} \\ m_{\mathrm{asc}} & \neg a_{21}\ \neg a_{22}\ \ldots\ \neg a_{2n} \end{pmatrix} \in \mathbb{B}^{2\times(1+n)}_{\mathrm{asc}}. \tag{29}$$

(i) We have $A \in \mathbb{B}^{2\times n}_{\mathrm{asc}}$ iff $m_{\mathrm{asc}} = 1$ is a solution for (28)–(29).

(ii) We have $A \in \mathbb{B}^{2\times n}_{\mathrm{desc}}$ iff $m_{\mathrm{asc}} = 0$ is a solution for (28)–(29).

Proof. (i) Suppose $A \in \mathbb{B}^{2\times n}_{\mathrm{asc}}$. Then, the choice $m_{\mathrm{asc}} = 1$ trivially satisfies (29), as the leftmost bit-comparison already entirely confirms the claimed lexicographic order. Condition (28) also applies when plugging in $m_{\mathrm{asc}} = 1$ (by the assumption on A). For the other proof direction we notice that as (28) just prepends a single comparison-tie and remains ascendingly lexicographically ordered, $A \in \mathbb{B}^{2\times n}_{\mathrm{asc}}$.

(ii) Suppose now $A \in \mathbb{B}^{2\times n}_{\mathrm{desc}}$. Then, trivially the choice $m_{\mathrm{asc}} = 0$ fulfills (28). Condition (29), for $m_{\mathrm{asc}} = 0$, implies that $(\neg a_{ij})_{i,j} \in \mathbb{B}^{2\times n}_{\mathrm{asc}}$ in turn being equivalent to $A \in \mathbb{B}^{2\times n}_{\mathrm{desc}}$.
Conversely, as for $m_{\mathrm{asc}} = 0$ the leftmost comparison in (29) results in a tie, we conclude $A \in \mathbb{B}^{2\times n}_{\mathrm{desc}}$.

Lemma 3. *A binary $n \times n$ matrix is bi-stochastic iff each row has* precisely one 1*-entry and each column has* at most one 1*-entry.*

Proof. Apply the pigeonhole principle to the columns.

Remark 4. A rough estimate of the *size of the encoding* in Sect. 2 can be obtained by the following observations: The d incidence matrices are represented by $dn(n-1)/2 = dO(n^2)$ decision variables. A number of $d\binom{n}{3}3! + d\sum_{4\le j\le k}\binom{n}{j}j = dO(n^k)$ additional variables signalizes occurrences of a semiordered pattern in each of the d permutations (assuming k is fixed); these have to be linked to the summands in (8)–(9). The number of constraints is dominated by the count of cardinality constraints to be installed for all of the latter summands, i.e., by $O(n^k)$ arithmetic inequalities which shall enforce that at most d/j variables out of a set of d variables shall attain the Boolean value 1; here a linear amount of auxiliary fresh variables to realize each counting constraint must typically be introduced; we refer to Sect. 4 for more specific information on how we realize such constraints. Given the latter magnitude of constraints, the count of $dO(n^3)$ transitivity-ensuring clauses, see (7), is negligible in comparison. A number of $d-1$ lexicographical comparisons have to be modeled; as we see in Sect. 4 their realization asks to introduce a number of auxiliary variables as large as the length of the binary strings representing the upper diagonal matrix, i.e., at most $(d-1)O(n^2)$ where the extremal case refers to the use of accuracy $L = n(n-1)/2 - 1$.

Supplementary material

The following are the certifying instances for Propositions 1, 3. Readers interested in inspecting these objects can find them below as `Julia` lists together with a standalone feasibility checker in Listing 1.2—the latter can be used as follows:

```
F = [apply_perm(offset,g) for offset in theta for g in G]
k=5; is_mw_indep(F, k)
```

Listing 1.1. Certifiers

```
#d,n,k;|G|=60,6,6;6
θ = [[2,4,5,3,6,1],[2,5,1,4,6,3],[4,6,1,3,2,5],[2,6,1,5,3,4],[3,2,1,4,5,6],
     [5,4,1,2,3,6],[4,3,1,5,6,2],[6,5,2,3,4,1],[4,3,2,5,1,6],[6,3,2,1,5,4]] # |θ|=10
G = [[1,2,3,4,5,6],[5,4,2,3,6,1],[6,3,4,2,1,5],[4,5,6,1,2,3],[2,1,5,6,3,4],[3,6,1,5,4,2]] # |G|=6

#d,n,k;|G|=60,6,5;6
θ = [[4,5,6,1,2,3],[3,4,5,2,6,1],[1,3,6,2,5,4],[1,3,6,4,2,5],[2,4,6,3,1,5],
     [1,2,6,5,4,3],[2,5,3,6,1,4],[3,6,4,5,1,2],[2,3,1,6,4,5],[2,5,3,1,6,4]] # |θ|=10
G = [[1,2,3,4,5,6],[5,4,2,3,6,1],[6,3,4,2,1,5],[4,5,6,1,2,3],[2,1,5,6,3,4],[3,6,1,5,4,2]] # |G|=6

#d,n,k;|G|=60,5,5;60
θ = [[1,4,3,2,5]] # |θ|=1
G = [[1,2,3,4,5],[1,4,2,3,5],[1,3,4,2,5],[1,4,3,5,2],[1,5,4,3,2],[1,3,5,4,2],
     [1,5,3,2,4],[1,2,5,3,4],[1,3,2,5,4],[1,5,2,4,3],[1,4,5,2,3],[1,2,4,5,3],
     [2,1,5,4,3],[2,4,1,5,3],[2,5,4,1,3],[2,4,5,3,1],[2,3,4,5,1],[2,5,3,4,1],
     [2,3,5,1,4],[2,1,3,5,4],[2,5,1,3,4],[2,3,1,4,5],[2,4,3,1,5],[2,1,4,3,5],
     [4,1,2,5,3],[4,5,1,2,3],[4,2,5,1,3],[4,5,2,3,1],[4,3,5,2,1],[4,2,3,5,1],
     [4,3,2,1,5],[4,1,3,2,5],[4,2,1,3,5],[4,3,1,5,2],[4,5,3,1,2],[4,1,5,3,2],
     [5,1,4,2,3],[5,2,1,4,3],[5,4,2,1,3],[5,2,4,3,1],[5,3,2,4,1],[5,4,3,2,1],
     [5,3,4,1,2],[5,1,3,4,2],[5,4,1,3,2],[5,3,1,2,4],[5,2,3,1,4],[5,1,2,3,4],
     [3,2,4,1,5],[3,1,2,4,5],[3,4,1,2,5],[3,1,4,5,2],[3,5,1,4,2],[3,4,5,1,2],
     [3,5,4,2,1],[3,2,5,4,1],[3,4,2,5,1],[3,5,2,1,4],[3,1,5,2,4],[3,2,1,5,4]] # |G|=60

#d,n,k;|G|=36,8,4;2
```

```
G = [[1,2,3,4,5,6,7,8],[8,7,6,5,4,3,2,1]] # |G|=2, exceptionally here considered as right-coset
θ = [[1,2,3,4,5,6,7,8],[8,5,6,7,2,3,4,1],[1,6,7,3,2,4,8,5],
     [5,2,4,8,6,7,3,1],[1,4,6,5,8,3,2,7],[7,8,3,2,4,6,5,1],
     [5,1,8,7,3,4,2,6],[6,3,4,2,1,8,7,5],[5,1,8,3,6,4,7,2],
     [2,6,4,7,1,8,3,5],[2,6,1,5,8,4,7,3],[3,8,4,7,6,1,5,2],
     [6,2,1,5,4,3,8,7],[7,4,3,8,2,1,5,6],[5,7,1,3,6,4,2,8],
     [8,6,4,2,7,1,3,5],[4,3,8,1,7,6,2,5],[5,7,6,2,3,8,1,4]] # |θ|=18, inferred from Na et al. 2023, Proposition 4.6(iii)

#d,n,k;|G|=48,8,4;24
θ = [[5,8,7,2,3,6,1,4],[6,5,1,4,2,3,8,7]] # |θ|=2
G = [[1,2,3,4,5,6,7,8],[1,5,7,2,4,3,6,8],[1,4,6,5,2,7,3,8],[8,7,4,3,6,5,2,1],
     [8,6,2,7,3,4,5,1],[8,3,5,6,7,2,4,1],[3,6,8,1,2,7,5,4],[3,2,5,6,1,8,7,4],
     [3,1,7,2,6,5,8,4],[4,5,1,8,7,2,6,3],[4,7,6,5,8,1,2,3],[4,8,2,7,5,6,1,3],
     [6,4,2,7,1,8,3,5],[6,1,3,4,7,2,8,5],[6,7,8,1,4,3,2,5],[5,3,7,2,8,1,4,6],
     [5,8,4,3,2,7,1,6],[5,2,1,8,3,4,7,6],[2,8,5,6,4,3,1,7],[2,4,1,8,6,5,3,7],
     [2,6,3,4,8,1,5,7],[7,1,6,5,3,4,8,2],[7,3,8,1,5,6,4,2],[7,5,4,3,1,8,6,2]] # |G|=24

#d,n,k;|G|=24,7,4;12
θ = [[7,2,5,6,3,4,1],[7,6,4,3,1,5,2]] # |θ|=2
G = [[1,2,3,4,5,6,7],[1,2,3,5,4,7,6],[3,2,1,5,4,6,7],[3,2,1,4,5,7,6],
     [6,2,7,3,1,5,4],[6,2,7,1,3,4,5],[7,2,6,1,3,5,4],[7,2,6,3,1,4,5],
     [5,2,4,7,6,1,3],[5,2,4,6,7,3,1],[4,2,5,6,7,1,3],[4,2,5,7,6,3,1]] # |G|=12

#d,n,k;|G|=24,6,4;24
θ = [[2,3,5,1,6,4]] # |θ|=1
G = [[1,2,3,4,5,6],[2,1,3,4,6,5],[5,6,3,4,1,2],[6,5,3,4,2,1],[1,2,4,3,6,5],[2,1,4,3,5,6],
     [6,5,4,3,1,2],[5,6,4,3,2,1],[3,4,5,6,1,2],[4,3,5,6,2,1],[1,2,5,6,3,4],[2,1,5,6,4,3],
     [3,4,6,5,2,1],[4,3,6,5,1,2],[2,1,6,5,3,4],[1,2,6,5,4,3],[5,6,1,2,3,4],[6,5,1,2,4,3],
     [3,4,1,2,5,6],[4,3,1,2,6,5],[5,6,2,1,4,3],[6,5,2,1,3,4],[4,3,2,1,5,6],[3,4,2,1,6,5]] # |G|=24

#d,n,k;|G|=12,5,4;6
θ = [[1,4,2,3,5],[5,2,4,3,1]] # |θ|=2
G = [[1,2,3,4,5],[1,3,2,5,4],[2,3,1,4,5],[2,1,3,5,4],[3,1,2,4,5],[3,2,1,5,4]] # |G|=6

#d,n,k;|G|=12,4,4;12
θ = [[1,3,4,2]] # |θ|=1
G = [[1,2,3,4],[1,3,4,2],[1,4,2,3],[4,3,2,1],[4,2,1,3],[4,1,3,2],
     [3,4,1,2],[3,1,2,4],[3,2,4,1],[2,1,4,3],[2,4,3,1],[2,3,1,4]] # |G|=12
```

Listing 1.2. Feasibility checker

```
using Combinatorics

"""
Returns the composition of two permutations ‘outer‘ and ‘inner‘ which both are passed as lists.
"""
function apply_perm(outer, inner)
   return [outer[inner[j]] for j in 1:length(inner)]
end

"""
Returns the union over all k in ‘k_range‘ of all sets of semiordered patterns SOP(n,k).
"""
function gen_SOP_n_k_range(k_range, n)
   SOP = []
   for j in k_range
      tmp_cll = []
      for cmb_j in combinations(1:n, j)
         for idx in 1:length(cmb_j)
            copy_cmb_j = copy(cmb_j)
            el = cmb_j[idx]
            deleteat!(copy_cmb_j, idx)
            push!(tmp_cll, tuple([el; sort(copy_cmb_j)]...))
         end
      end
      sort!(tmp_cll)
      append!(SOP, tmp_cll)
   end
   return SOP
end

"""
Checks if the family ‘F‘ is ‘k‘-restricted minwise independent.
"""
function is_mw_indep(F, k)
   SOP = gen_SOP_n_k_range(2:k, length(F[1]))
   unmet = []
   is_valid = true
   for sop in SOP
      counter = 0
      for l in 1:length(F)
         if all([F[l][sop[1]] < F[l][sop[j]] for j in 2:length(sop)])
            counter += 1
         end
      end
      if counter != div(length(F), length(sop))
         push!(unmet, (sop, counter))
         is_valid = false
      end
```

```
  end
  is_valid == false && println("unmet multiplicity: ", unmet)
  return is_valid
end
```