1 Notation

Let A be some set. We will denote the set $\{\{a_1, \dots, a_k\} \subseteq A : a_1, \dots, a_k \text{ pairwise distinct}\}$, which contains all k-ary subsets of A, by $\binom{A}{k}$. For the same set A, P_A denotes the set of all partitions of A. For a partition $\Pi \in P_A$ and an element $a \in A$, $[a]_\Pi$ denotes the set in Π that contains a, of which there is exactly one. If A is finite, $\mathcal{U}(A)$ denotes the uniform distribution over A, i.e. the distribution that assigns every element in A the same probability (namely: $1/|A|$). If we want to indicate that a random variable $\mathbf{X}$ has probability distribution $\mathcal{Q}$, we will write $\mathbf{X} \sim \mathcal{Q}$. If f is a function from the domain of a random variable $\mathbf{X} \sim \mathcal{Q}$ to the real numbers, its expected value is written as $\mathbb{E}_{\mathbf{X}\sim\mathcal{Q}}[f(\mathbf{X})]$.

2 Introduction

Let $V = \{v_1, \dots, v_n\}$ be a finite set of vertices. Associated with this set are functions $c, c' : \binom{V}{3} \mapsto \mathbb{R}$ which define cost-structures on 3-ary subsets of V. For a 3-ary subset $\{u, v, w\} \in \binom{V}{3}$ and a given partition $\Pi \in P_V$, we define the cost of $\{u, v, w\}$ with respect to Π as

$$\ell(\{u, v, w\}, \Pi) = \begin{cases} c(\{u, v, w\}) & \text{if } [u]_\Pi \neq [v]_\Pi, [u]_\Pi \neq [w]_\Pi, [w]_\Pi \neq [v]_\Pi \\ c'(\{u, v, w\}) & \text{if } [u]_\Pi = [v]_\Pi = [w]_\Pi \\ 0 & \text{otherwise.} \end{cases}$$

This can be interpreted as follows: whenever u, v and w are part of pairwise different sets in Π, the cost of $\{u, v, w\}$ is equal to the costs as defined by c. If they are part of the same set, the cost of $\{u, v, w\}$ is equal to the costs as defined by c'. Otherwise, if neither of the above is the case, the cost of $\{u, v, w\}$ is just 0. Based on this definition, we are confronted with problems of the form

$$\Pi^* = \arg\min_{\Pi \in P_V} \sum\nolimits_{\{u,v,w\} \in \binom{V}{3}} \ell(\{u, v, w\}, \Pi)$$

i.e. we wish to find a partition Π^* of V that minimizes some objective function over 3-ary subsets of V. This problem is hard in many respects: one, the amount of possible partitions grows large very fast (with growing n). Two, even computing the objective function for a given partition is prohibitive, since summing over all 3-ary subsets takes almost n^3 steps (and to add to that, not even n^2 steps are feasible if n becomes larger). Thus, we are interested in a way of approximately solving this problem by the use of a local search algorithm and some considerations on the objective function.

Approximizing the Objective Function Since we are only interested in some minimizer Π^* of the above problem, it does not matter which function we minimize, as long as the set of minimizers stays the same. Therefore, we can apply any strongly monotonic growing function to the objective, and, for example, multiply by some constant. We are then able to obtain

$$\begin{aligned} \Pi^* &= \arg\min_{\Pi \in P_V} \sum\nolimits_{T \in \binom{V}{3}} \ell(T, \Pi) \\ &= \arg\min_{\Pi \in P_V} \frac{1}{|\binom{V}{3}|} \sum\nolimits_{T \in \binom{V}{3}} \ell(T, \Pi) \\ &= \arg\min_{\Pi \in P_V} \mathbb{E}_{\mathbf{T} \sim \mathcal{U}(\binom{V}{3})} [\ell(\mathbf{T}, \Pi)] \end{aligned}$$

where $\mathcal{U}(\binom{V}{3})$ is the uniform distribution over $\binom{V}{3}$. This perspective allows to approximate the objective value to any degree by uniformly sampling a fixed number of 3-ary subsets from V

and then computing the sample mean. In many cases, one wants to compute the change in the objective function when considering two different partitions $\Pi, \Pi' \in P_V$. The difference is then given by

$$\mathbb{E}_{\mathbf{T}\sim\mathcal{U}(\binom{V}{3})}\left[\ell(\mathbf{T},\Pi)\right] - \mathbb{E}_{\mathbf{T}\sim\mathcal{U}(\binom{V}{3})}\left[\ell(\mathbf{T},\Pi')\right] = \mathbb{E}_{\mathbf{T}\sim\mathcal{U}(\binom{V}{3})}\left[\ell(\mathbf{T},\Pi) - \ell(\mathbf{T},\Pi')\right],$$

which can possibly be simplified, dependent on the shape of Π' with respect to Π and vice versa.

3 Indexings as Proxies for Partitions

Since we want to use a local search algorithm that iteratively finds better solutions (partitions in this case), the second consideration is on how we want to represent said partitions. One may represent a partition as a set of sets of vertices, which is probably the most straight-forward approach. On the one hand, the required space is linear in n. But on the other hand, this comes with a few problems: checking if two vertices are part of the same set, and the removal and the addition of a vertex from or to a set all require n steps in the worst case (if all vertices have the same set). A last problem is that one has to gurantee consistency at every step, i.e. that every solution is indeed a partition. A second idea might be to store a partition as a $n \times n$ boolean matrix, where every row corresponds to a vertex and every column to a set in the partition, and an entry at position i, j would mean "vertex i is part of set j" if 1 and "vertex i is not part of set j" if 0. This would alleviate some of the above problems, since looking up, removal or addition of vertices to sets would take constant time in the worst case. However, storing a partition would require n^2 units of space, which might become problematic when n is large or when considering multiple partitions at once.

Therefore, we will use the following concept of indexings (which are similar to equivalence relations) to solve the above problems, i.e. to simplify and speed up operations on partitions. Every vertex is mapped to a number (index) which indicates the set in the partition we want this vertex to be a part of. If multiple vertices are mapped to the same index, they will be part of the same set in the partition. Since there are at maximum n sets in a partition of V (every vertex is assigned its own set, i.e. the indexing is bijective), we restrict to indices between 1 and n. Note that every indexing can be stored in a linear amount of memory dependent on the size of V.

Definition 3.1. *Let $V = \{v_1, v_2, \ldots, v_n\}$ be a set of n vertices. An indexing of V is a mapping $\varphi \in \{1, \ldots, n\}^V = [n]^V$ that associates every vertex with a number from 1 to n.*

Definition 3.2. *Let φ be an indexing of V. The partition induced by φ is defined as*

$$\Pi(\varphi) = \{[v]_\varphi : v \in V\}, \tag{3.1}$$

where $[v]_\varphi = \{w \in V : \varphi(w) = \varphi(v)\}$.

Based on this definition, we are able to obtain some immediate results (proofs in the appendix). First, we are able to use indexings as representations for partitions, i.e. for every partition, there is some indexing that yields that very partition (Lemma 3.1). And two, we obtain a criteria that makes it possible to check when two vertices share the same set in the partition (Lemma 3.2).

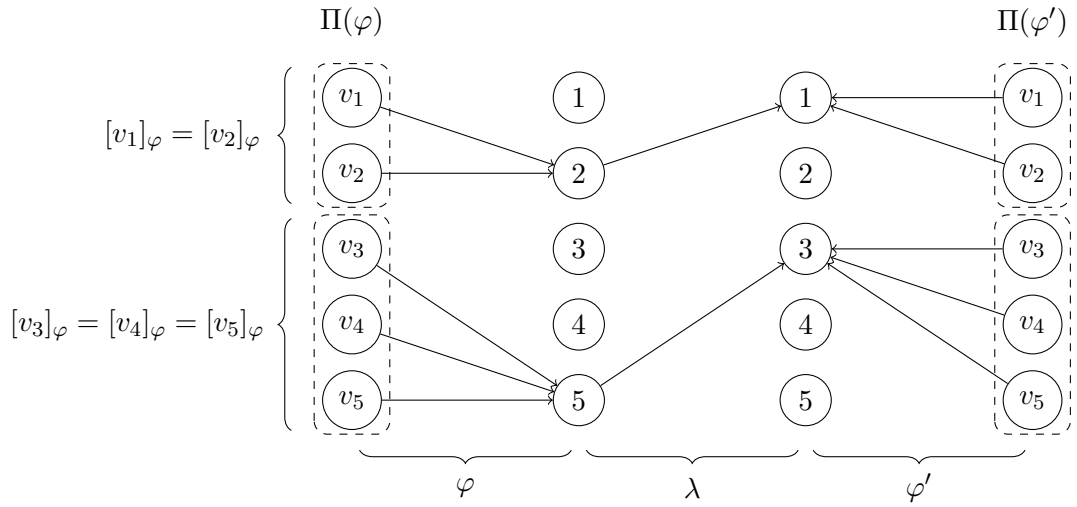

Figure 1: Illustration of two indexings φ, φ' with corresponding partitions and proof of equality through part (b) of Theorem 3.3.

Lemma 3.1. *Π is a partition of V if and only if there exists an indexing of V that induces Π.*

Lemma 3.2. *Let φ be an indexing of V. For all vertices v, u, we have $\varphi(v) = \varphi(u)$ if and only if v and u are part of the same set in $\Pi(\varphi)$.*

Since we want to use indexings in order to define transformations on partitions, we are interested in the question when two indexings are "equal", in the sense that they induce the same partition. This is characterized in part by Theorem 3.3.

Theorem 3.3. *If φ, φ' are two indexings of V, then the following statements are equivalent:*

(a) $\Pi(\varphi) = \Pi(\varphi')$

(b) there is a bijection $\lambda : \text{image}(\varphi) \to \text{image}(\varphi')$[1] *such that* $\lambda(\varphi(v)) = \varphi'(v)$ *for all* $v \in V$

(c) for all vertices v, w, $\varphi(w) = \varphi(v)$ *if and only if* $\varphi'(w) = \varphi'(v)$

If the task is to determine whether two induced partitions $\Pi(\varphi), \Pi(\varphi')$ for given indexings φ, φ' of V are equal, the result of Theorem 3.3 might be useful: instead of explicitly computing the resulting partitions and checking if both sets are equal, one can simply determine whether a fitting bijection λ exists, which is arguably easier. In fact, one can construct λ as in the first part of the proof of Lemma 3.3 and then check whether the result is a bijection or not. Figure 1 illustrates two indexings, their partitions and some of the findings of Theorem 3.3.

4 Move-Operation on Indexings

We will now define the "move"-operation on indexings, which transforms an indexing into another indexing by changing the assigned index of a vertex. The effect this has on the induced partition is that the respective vertex is "taken" from its original set[2] and then "put into" some

[1] $\text{image}(\varphi)$ means the image of φ, i.e. $\text{image}(\varphi) = \{\varphi(v)\}_{v \in V}$.

[2] i.e. the set $[v]_\varphi$

other set[3]. This is then defined as $\mathrm{move} : [n]^V \times V \times [n] \to [n]^V$ with

$$\mathrm{move}(\varphi, v, k)(u) = \begin{cases} \varphi(u) & u \neq v, \\ k & u = v \end{cases} \tag{4.1}$$

for every vertex u. The operation takes as input an indexing φ, a vertex v and new index k and outputs a new indexing that is essentially the same as φ, with the only difference being that v is mapped to k instead of whatever it was mapped to before.

In many cases, there are multiple ways of moving one vertex to different indices while still inducing the same partition afterwards. For example, if v is an arbitrary vertex, φ is an indexing of V with $\varphi^{-1}(k_1) = \cdots = \varphi^{-1}(k_m) = \emptyset$ and $k_1, \ldots, k_m$ are pairwise different, then moving v to $k_1, \ldots, k_m$ yields that while $\mathrm{move}(\varphi, v, k_1), \ldots, \mathrm{move}(\varphi, v, k_m)$ are all different indexings, their induced partitions are the same: $\Pi(\mathrm{move}(\varphi, v, k_1)) = \cdots = \Pi(\mathrm{move}(\varphi, v, k_m))$.
Based on this observation we are interested in an efficient way of enumerating all possible "moves" of a vertex with respect to the induced partitions without enumerating too much or having to double-check whether two move-operations induce the same partition. Hence, consider algorithm 1, which aims at finding a solution to this problem.

Algorithm 1: Move-Enumeration

Input: Set of vertices V with indexing φ
Result: Sequence of indexings $\varphi_1, \varphi_2, \ldots, \varphi_m$

```
Let < be some linear order on V ;
Let 𝒩 := {1, ..., n}\image(φ) ;
forall vertices w ∈ V do
    forall φ(v) ∈ image(φ)\{φ(w)} do
        if [w]_φ = {w, u, ...} or [v]_φ = {v, s, ...} then
            enumerate move(φ, w, φ(v)) ;
        else if [w]_φ = {w} and [v]_φ = {v} and w < v then
            enumerate move(φ, w, φ(v)) ;
    if [w]_φ = {w, u, v, ...} or ([w]_φ = {w, u} and w < u) then
        Let k ∈ 𝒩 ;
        enumerate move(φ, w, k) ;
```

We want to check three important properties: one, algorithm 1 only enumerates indexings which yield partitions distinct from all induced partitions of the other indexings (the "not too much"-part, see Lemma 4.1), two, every partition that is induced when moving one vertex in φ to a different set is induced by some indexing in $\varphi_1, \ldots, \varphi_m$ (Lemma 4.2) and three, the partition that is induced by $\mathrm{move}(\varphi, v, k)$ for any vertex v and number k is not the same that is induced by φ (Lemma 4.3).

Lemma 4.1 (Pairwise Distinctiveness)**.** *Let φ be an indexing of V and $\varphi_1, \ldots, \varphi_m$ be a sequence generated by algorithm 1 on input V and φ. Then for all pairwise distinct $i, j \in \{1, \ldots, m\}$, we have $\Pi(\varphi_i) \neq \Pi(\varphi_j)$.*

Lemma 4.2 (Completeness)**.** *Let φ be an indexing of V and $\varphi_1, \ldots, \varphi_m$ be the sequence that is generated by algorithm 1 on input V and φ. For all vertices v and $k \in \{1, \ldots, n\}$ such that $\Pi(\mathrm{move}(\varphi, v, k)) \neq \Pi(\varphi)$, there is $i \in \{1, \ldots, m\}$ such that $\Pi(\mathrm{move}(\varphi, v, k)) = \Pi(\varphi_i)$.*

[3] i.e. the set $\varphi^{-1}(k)$, where k is some number between 1 and n

Lemma 4.3. *Let φ be an indexing of V and $\varphi_1, \ldots, \varphi_m$ be the sequence that is generated by algorithm 1 on input V and φ. Then $\Pi(\varphi) \neq \Pi(\varphi_i)$ for all $i \in \{1, \ldots, m\}$.*

Corollary 4.3.1. *For an indexing ψ of V, the following statements are equivalent:*

- $\psi = \mathrm{move}(\varphi, v, k)$ *and* $\Pi(\psi) \neq \Pi(\varphi)$ *for* $v \in V, 1 \leq k \leq n$
- $\Pi(\psi) \in \{\Pi(\varphi_1), \ldots, \Pi(\varphi_m)\}$.

Proof. From top to bottom, simply apply Lemma 4.2. For the direction from bottom to top, apply Lemma 4.3. □

A Proofs for Section 3 (Indexings as Proxies for Partitions)

Lemma 3.1. *Π is a partition of V if and only if there exists an indexing of V that induces Π.*

Proof. First, we will show the direction from left to right. Let $\Pi = \{U_1, \ldots, U_m\}, 1 \leq m \leq n$, be a partition of V. For all vertices v with corresponding set U_i in Π (of which there is exactly one), we define $\varphi(v) = i$. For a $j \in \{1, \ldots, m\}$ we then obtain $\varphi^{-1}(j) = U_j$. But then $\Pi(\varphi) = \Pi$, i.e. Π is induced by φ.
The other direction requires us to prove that every indexing induces a partition of V. I.e., for an indexing φ of V with $\Pi(\varphi) = \Pi$, we need to check the following requirements:

1. $\emptyset \notin \Pi$: This is obvious from (3.1), since every element $[v]_\varphi \in \Pi$ contains at least v.
2. $\bigcup_{U \in \Pi} U = V$: "$\subseteq$" is obvious. For "$\supseteq$", take a vertex v. Then, $v \in [v]_\varphi$ and since $[v]_\varphi \in \Pi(\varphi)$, we obtain $v \in \bigcup_{U \in \Pi} U$.
3. if $U_1, U_2 \in \Pi$ and $U_1 \neq U_2$, then $U_1 \cap U_2 = \emptyset$: Take two $U_1, U_2 \in \Pi$ with $U_1 \neq U_2$ such that $U_1 = [w]_\varphi$ and $U_2 = [u]_\varphi$ for two vertices w and u. Assume for a contradiction that $U_1 \cap U_2 \neq \emptyset$, i.e. there exists $v \in U_1 \cap U_2$. But then $v \in [w]_\varphi$ and $v \in [u]_\varphi$. Thus, $\varphi(v) = \varphi(w) = \varphi(u)$, which cannot be the case, since that would imply $U_1 = U_2$.

This completes the proof. □

Lemma 3.2. *Let φ be an indexing of V. For all vertices v, u, we have $\varphi(v) = \varphi(u)$ if and only if v and u are part of the same set in $\Pi(\varphi)$.*

Proof. For the direction from left to right, take $v, u \in V$ such that $\varphi(v) = \varphi(u)$. Then $v, u \in [v]_\varphi = [u]_\varphi \in \Pi(\varphi)$, i.e. they are part of the same set. For the other direction, we will show the contraposition. Take $v, u \in V$ such that $\varphi(v) \neq \varphi(u)$. Clearly, $v \in [v]_\varphi$ and $v \notin [u]_\varphi$ as well as $u \in [u]_\varphi$ and $u \notin [v]_\varphi$. Therefore $[v]_\varphi \neq [u]_\varphi$. Since by Lemma 3.1, $\Pi(\varphi)$ is a partition and $[v]_\varphi$ and $[u]_\varphi$ are distinct sets in $\Pi(\varphi)$, $[v]_\varphi$ is the only set that contains v and vice versa for $[u]_\varphi$ and u. Thus, v and u cannot be part of the same set in $\Pi(\varphi)$. □

Theorem 3.3. *If φ, φ' are two indexings of V, then the following statements are equivalent:*

(a) $\Pi(\varphi) = \Pi(\varphi')$

(b) there is a bijection $\lambda : \mathrm{image}(\varphi) \to \mathrm{image}(\varphi')$[4] *such that* $\lambda(\varphi(v)) = \varphi'(v)$ *for all* $v \in V$

(c) for all vertices v, w, $\varphi(w) = \varphi(v)$ *if and only if* $\varphi'(w) = \varphi'(v)$

Proof. We will start with the direction from (a) to (b). Let φ, φ' be two indexings of V with $\Pi(\varphi) = \Pi(\varphi')$. We define $\lambda : \mathrm{image}(\varphi) \to \mathrm{image}(\varphi')$ as follows. For all $k \in \mathrm{image}(\varphi)$, associate a vertex v such that $\varphi(v) = k$. Then define $\lambda(k) = \varphi'(v)$. It remains to show that λ fulfills the requirements in (b):

1. λ is bijective: Since every element in $\mathrm{image}(\varphi)$ corresponds to a set in $\Pi(\varphi)$, and vice versa for $\mathrm{image}(\varphi')$ and $\Pi(\varphi')$, we get $|\mathrm{image}(\varphi)| = |\Pi(\varphi)| = |\Pi(\varphi')| = |\mathrm{image}(\varphi')|$. Since $\mathrm{image}(\varphi)$ and $\mathrm{image}(\varphi')$ are also finite, it suffices to show injectivity of λ in order to prove bijectivity. Assume for a contradiction that λ is not injective, i.e. there are $v_1, v_2 \in V, \varphi(v_1) \neq \varphi(v_2)$, such that $\lambda(\varphi(v_1)) = \lambda(\varphi(v_2)) = \varphi'(v_1) = \varphi'(v_2)$. Thus, by

[4] $\mathrm{image}(\varphi)$ means the image of φ, i.e. $\mathrm{image}(\varphi) = \{\varphi(v)\}_{v \in V}$.

application of Lemma 3.2, there must be a set in $\Pi(\varphi')$ that contains v_1 and v_2, while there is no set in $\Pi(\varphi)$ that has both vertices in it. But then $\Pi(\varphi) \neq \Pi(\varphi')$. Contradiction.

2. For all vertices v, $\lambda(\varphi(v)) = \varphi'(v)$: Take an arbitrary vertex v and let $\varphi(v) = k \in \mathrm{image}(\varphi)$. Let $\bar{v}$ be the vertex that was previously associated with k in the definition of λ, i.e. the vertex for which $\varphi(\bar{v}) = k = \varphi(v)$ and $\lambda(\varphi(\bar{v})) = \varphi'(\bar{v})$ holds. Assume for a contradiction that $\varphi'(v) \neq \varphi'(\bar{v})$. By Lemma 3.2, this yields that there is no set in $\Pi(\varphi')$ that contains both v and $\bar{v}$, while there is one in $\Pi(\varphi)$. But then again $\Pi(\varphi) \neq \Pi(\varphi')$, i.e. a contradiction. Thus, $\lambda(\varphi(v)) = \lambda(\varphi(\bar{v})) = \varphi'(\bar{v}) = \varphi'(v)$.

This concludes this direction. For the direction from (b) to (c), let λ be a bijection between the images of two indexings φ, φ' of V that fulfills the requirements in (b). Let w, v be two arbitrary vertices; then

$$\begin{aligned} \varphi(w) = \varphi(v) \quad &\text{iff} \quad \lambda(\varphi(w)) = \lambda(\varphi(v)) && (*) \\ &\text{iff} \quad \varphi'(w) = \varphi'(v) && (**) \end{aligned}$$

The first identity $(*)$ works since λ is a bijection and the second $(**)$ since $\lambda(\varphi(w)) = \varphi'(w)$ for all vertices w. The remainder (c) to (a) is relatively trivial: If we assume premise (c), then

$$\begin{aligned} U \in \Pi(\varphi) \quad &\text{iff} \quad U = [v]_\varphi \text{ for a vertex } v \\ &\text{iff} \quad U = \{w \in V : \varphi(w) = \varphi(v)\} \text{ for a vertex } v \\ &\text{iff} \quad U = \{w \in V : \varphi'(w) = \varphi'(v)\} \text{ for a vertex } v \\ &\text{iff} \quad U = [v]_{\varphi'} \text{ for a vertex } v \\ &\text{iff} \quad U \in \Pi(\varphi'), \end{aligned}$$

which shows $\Pi(\varphi) = \Pi(\varphi')$. □

B Proofs for Section 4 (Move-Operation on Indexings)

Lemma 4.1 (Pairwise Distinctiveness). *Let φ be an indexing of V and $\varphi_1, \ldots, \varphi_m$ be a sequence generated by algorithm 1 on input V and φ. Then for all pairwise distinct $i, j \in \{1, \ldots, m\}$, we have $\Pi(\varphi_i) \neq \Pi(\varphi_j)$.*

Proof. First note that $\varphi_1, \ldots, \varphi_m$ are indeed results of a move-operation on φ. This can be checked by simple introspection on algorithm 1. Therefore, we can write $\varphi_i = \mathrm{move}(\varphi, w_i, k_i)$ for all $i = 1, \ldots, m$, with $w_i \in V$ and $k_i \in \{1, \ldots, n\}$, i.e.: φ_i moves vertex w_i to index k_i. Secondly, note that every pair $(w, k) \in V \times \{1, \ldots, n\}$ is regarded at most once. This means that for every two different indexings φ_i, φ_j, either $w_i \neq w_j$ or $k_i \neq k_j$ holds. Now, pick $i, j \in \{1, \ldots, m\}$ such that $i \neq j$. The rest of the proof can be done via case distinction on all possible conditions under which a move-operation could be executed in the algorithm:

1. $w_i = w_j = w$. Since i and j are pairwise distinct, $k_i \neq k_j$ must hold. For k_i, we have either $k_i \in \mathcal{N}$ (line 10) or $k_i = \varphi(v) \in \mathrm{image}(\varphi) \backslash \{\varphi(w)\}$ (line 6 and 8):

   (a) If $k_i \in \mathcal{N}$: Then $\varphi_i(w) = k_i$ only for w (since there is no vertex that is mapped to k_i in φ). Also $k_j \notin \mathcal{N}$, since there is at maximum one $k \in \mathcal{N}$ for which $\mathrm{move}(\varphi, w, k)$ is enumerated. Thus, $k_j = \varphi(v) \in \mathrm{image}(\varphi) \backslash \{\varphi(w)\}$. But then we have $\varphi_j(w) = k_j = \varphi_j(v)$ and $\varphi_i(v) = k_j \neq k_i = \varphi_i(w)$, which implies $\Pi(\varphi_i) \neq \Pi(\varphi_j)$ by Theorem 3.3.

(b) If $k_i = \varphi(v) \in \text{image}(\varphi)\backslash\{\varphi(w)\}$: After moving w to k_i in φ_i, one obtains $\varphi_i(w) = \varphi_i(v) = k_i$, and after moving w to k_j in φ_j, one gets $\varphi_j(w) = k_j \neq k_i = \varphi_j(v)$. But that implies $\Pi(\varphi_i) \neq \Pi(\varphi_j)$ by Theorem 3.3.

2. $w_i \neq w_j$, and therefore $\varphi_i(w_j) = \varphi(w_j)$ and $\varphi_j(w_i) = \varphi(w_i)$ (i.e., φ_i does not change the index of w_j and φ_j does not change the index of w_i). Again, we make the distinction for the case $k_i \in \mathcal{N}$ and $k_i = \varphi(v) \in \text{image}(\varphi)\backslash\{\varphi(w)\}$:

   (a) If $k_i \in \mathcal{N}$: since $k_i \notin \text{image}(\varphi)$, $\varphi(w_i) \neq k_i$. Also, at least one of the following cases (see line 9 of the algorithm) must hold:

      i. If $[w_i]_\varphi = \{w_i, u, v, \dots\}$: At least one of the vertices u and v must be different from w_j, since $u \neq v$ and w_j cannot be equal to both of them. Let w.l.o.g. $u \neq w_j$. Then $\varphi_j(w_i) = \varphi(w_i) = \varphi(u) = \varphi_j(u)$ but $\varphi_i(w_i) = k_i \neq \varphi(w_i) = \varphi_i(u)$, i.e. $\Pi(\varphi_i) \neq \Pi(\varphi_j)$ by Theorem 3.3.

      ii. If $[w_i]_\varphi = \{w_i, u\}$ and $w_i < u$:

         A. $w_j \neq u$: We get $\varphi_i(w_i) \neq \varphi_i(u)$ and $\varphi_j(w_i) = \varphi(w_i) = \varphi(u) = \varphi_j(u)$. Then simply apply Theorem 3.3 and obtain $\Pi(\varphi_i) \neq \Pi(\varphi_j)$.

         B. $w_j = u$. Since $u \not< w_i$ and $[w_j]_\varphi = [w_i]_\varphi$, there is no possibility that line 11 is executed for w_j. Therefore, $k_j \notin \mathcal{N}$. But then $k_j = \varphi(v) \in \text{image}(\varphi)\backslash\{\varphi(w_j)\}$ with $w_i \neq v \neq w_j$. Thus, $\varphi_j(w_j) = \varphi(v) = \varphi_j(v)$ and $\varphi_i(w_j) = \varphi(w_j) \neq \varphi(v) = \varphi_i(v)$. Again, the application of Theorem 3.3 yields $\Pi(\varphi_i) \neq \Pi(\varphi_j)$.

   (b) If $k_i = \varphi(v) \in \text{image}(\varphi)\backslash\{\varphi(w_i)\}$, then one of the following cases applies:

      i. $[w_i]_\varphi = \{w_i, u, \dots\}$.

         A. If $w_j = u$: then $\varphi_j(w_i) = \varphi(w_i) \neq \varphi(v) = \varphi_j(v)$ and $\varphi_i(w_i) = \varphi(v) = \varphi_i(v)$. Here, Theorem 3.3 can be applied, which results in $\Pi(\varphi_i) \neq \Pi(\varphi_j)$.

         B. If $w_j \neq u$: then $\varphi_j(w_i) = \varphi(w_i) = \varphi(u) = \varphi_j(u)$ and $\varphi_i(w_i) \neq \varphi(w_i) = \varphi(u) = \varphi_i(u)$. Theorem 3.3 can be applied with result $\Pi(\varphi_i) \neq \Pi(\varphi_j)$.

      ii. $[v]_\varphi = \{v, s, \dots\}$.

         A. If $w_j = v$: then $\varphi_i(w_i) = \varphi_i(v)$ but $\varphi_j(w_i) \neq \varphi_j(v)$. In this case, Theorem 3.3 can be applied to obtain $\Pi(\varphi_i) \neq \Pi(\varphi_j)$.

         B. If $w_j \neq v$: then $\varphi_i(w_i) = \varphi_i(v)$ but $\varphi_j(w_i) \neq \varphi_j(v)$. Application of Theorem 3.3 yields $\Pi(\varphi_i) \neq \Pi(\varphi_j)$.

      iii. $[w_i]_\varphi = \{w_i\}$ and $[v]_\varphi = \{v\}$ and $w_i < v$.

         A. If $w_j = v$. Then neither line 6 (since $|[w_j]_\varphi| = 1$) nor line 8 are executed for w_j and $k = \varphi(w_i)$ (since $w_j \not< w_i$). Thus, $k_j \neq \varphi(w_i)$ must hold. But then $\varphi_i(w_i) = k_i = \varphi(v) = \varphi(w_j) = \varphi_i(w_j)$ and $\varphi_j(w_i) = \varphi(w_i) \neq k_j = \varphi_j(w_j)$. Application of Theorem 3.3 yields $\Pi(\varphi_i) \neq \Pi(\varphi_j)$.

         B. If $w_j \neq v$. Then $\varphi_i(w_i) = \varphi(v) = \varphi_i(v)$ but $\varphi_j(w_i) = \varphi(w_i) \neq \varphi(v) = \varphi_j(v)$. Again, application of Theorem 3.3 gives us $\Pi(\varphi_i) \neq \Pi(\varphi')$.

This finishes the proof, since in all cases $\Pi(\varphi_i) \neq \Pi(\varphi_j)$ holds. □

Lemma 4.2 (Completeness). *Let φ be an indexing of V and $\varphi_1, \ldots, \varphi_m$ be the sequence that is generated by algorithm 1 on input V and φ. For all vertices v and $k \in \{1, \ldots, n\}$ such that $\Pi(\mathrm{move}(\varphi, v, k)) \neq \Pi(\varphi)$, there is $i \in \{1, \ldots, m\}$ such that $\Pi(\mathrm{move}(\varphi, v, k)) = \Pi(\varphi_i)$.*

Proof. Let φ be an indexing of V and let w be a vertex and $k \in \{1, \ldots, n\}$. We want to show that if $\Pi(\mathrm{move}(\varphi, w, k)) \neq \Pi(\varphi)$, then there is an $i \in \{1, \ldots, m\}$ such that $\Pi(\mathrm{move}(\varphi, w, k)) = \Pi(\varphi_i)$. First, note that $\varphi(w) \neq k$ holds, since otherwise this would imply $\Pi(\mathrm{move}(\varphi, w, k)) = \Pi(\varphi)$. The remainder of this proof works with multiple case distinctions:

1. If $[w]_\varphi = \{w\}$. This directly implies $k = \varphi(v)$ for a $v \in V$, since otherwise that would mean $\Pi(\mathrm{move}(\varphi, w, k)) = \Pi(\varphi)$. Thus, for $[v]_\varphi$ there are the following options:
   (a) If $[v]_\varphi = \{v\}$. In the case $w < v$, line 8 enumerates $\mathrm{move}(\varphi, w, k)$ directly. Otherwise, if $v < w$, line 8 enumerates $\mathrm{move}(\varphi, v, \varphi(w))$, where $\Pi(\mathrm{move}(\varphi, v, \varphi(w)) = \Pi(\mathrm{move}(\varphi, w, k))$.
   (b) If $[v]_\varphi = \{v, s, \ldots\}$. Line 6 in the algorithm directly enumerates $\mathrm{move}(\varphi, w, k)$.
2. If $[w]_\varphi = \{w, u, \ldots\}$. Since φ maps w and u to the same index, there is at least one index in $1, \ldots, n$ that is assigned no vertex. But then $\mathcal{N} \neq \emptyset$. Again, there are the following options:
   (a) There is no $v \in V$ such that $k = \varphi(v)$:
      i. If $[w]_\varphi = \{w, v, u, \ldots\}$, then line 11 is executed and there is some $\ell \in \mathcal{N}$ for which $\mathrm{move}(\varphi, w, \ell)$ is enumerated. But then $\Pi(\mathrm{move}(\varphi, w, k)) = \Pi(\mathrm{move}(\varphi, w, \ell))$.
      ii. If $[w]_\varphi = \{w, u\}$.
         A. If $w < u$, then line 11 enumerates $\mathrm{move}(\varphi, w, \ell)$ for w and some $\ell \in \mathcal{N}$. But then $\Pi(\mathrm{move}(\varphi, w, \ell)) = \Pi(\mathrm{move}(\varphi, w, k))$.
         B. If $u < w$, then line 11 enumerates $\mathrm{move}(\varphi, u, \ell)$ for u and some $\ell \in \mathcal{N}$. But then again, $\Pi(\mathrm{move}(\varphi, u, \ell)) = \Pi(\mathrm{move}(\varphi, w, k))$.
   (b) There is $v \in V$ such that $k = \varphi(v)$. Then $k \in \mathrm{image}(\varphi) \backslash \{\varphi(w)\}$ and line 6 is executed. This enumerates $\mathrm{move}(\varphi, w, k)$.

This shows that in all cases, there is some φ_i that is enumerated which yields the same partition as $\mathrm{move}(\varphi, v, k)$. □

Lemma 4.3. *Let φ be an indexing of V and $\varphi_1, \ldots, \varphi_m$ be the sequence that is generated by algorithm 1 on input V and φ. Then $\Pi(\varphi) \neq \Pi(\varphi_i)$ for all $i \in \{1, \ldots, m\}$.*

Proof. $\Pi(\mathrm{move}(\varphi, v, k)) = \Pi(\varphi)$ if and only if either $k = \varphi(v)$ or if $[v]_\varphi = \{v\}$, then $k \in \mathcal{N}$. Simple case distinction yields that both cases never happen for any φ_i, $i \in \{1, \ldots, m\}$. □