Constructing Clustering Transformations

Steffen Borgwardt* and Charles Viss†

Abstract. Clustering is one of the fundamental tasks in data analytics and machine learning. In many situations, different clusterings of the same data set become relevant. For example, different algorithms for the same clustering task may return dramatically different solutions. We are interested in applications in which one clustering has to be transformed into another; e.g., when a gradual transition from an old solution to a new one is required. In this paper, we devise methods for constructing such a transition based on linear programming and network theory. We use a so-called clustering-difference graph to model the desired transformation and provide methods for decomposing the graph into a sequence of elementary moves that accomplishes the transformation. These moves are equivalent to the edge directions, or circuits, of the underlying partition polytopes. Therefore, in addition to a conceptually new metric for measuring the distance between clusterings, we provide new bounds on the circuit diameter of these partition polytopes.

Key words. partitioning, clustering, polyhedra, circuits, diameter, linear programming

AMS subject classifications. 52B05, 90C05, 90C08, 90C27

1. Introduction and Preliminaries.

Clusterings of large data sets play an important role in data analytics, machine learning, and informed decision-making in general. In many applications, there exists a desired clustering corresponding to an optimal solution to an optimization problem. However, directly implementing such a solution can be challenging – instead, a gradual sequence of transitions which transforms an initial, sub-optimal solution into the improved clustering is desired.

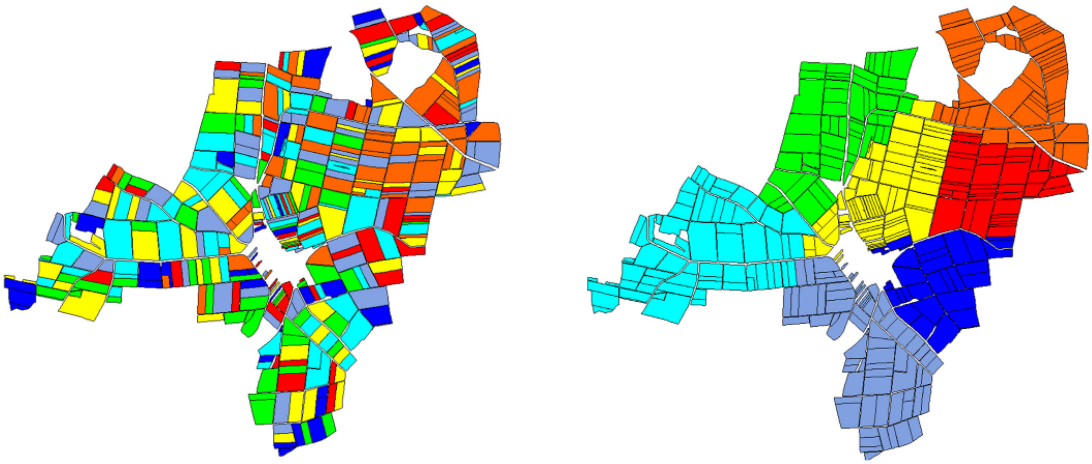
Consider the example in land consolidation from [3, 4, 5]. In a Bavarian agricultural region, 471 lots are cultivated by 7 different farmers. The initial distribution of the ownership of the lots, depicted in Figure 1a, is quite problematic – the scattered, small lots result in large transportation overhead and prohibit the use of heavy machinery. To address this, the authors worked with the Bavarian State to facilitate a voluntary land exchange among the farmers in which the boundaries of the lots would remain the same while the cultivation rights for the lots would be redistributed.

The combinatorial redistribution of lots can be modeled as a clustering problem: a set of data points (the lots) must be partitioned into clusters (the farmers) under the restriction that each farmer stays close to his original total value of land. The original and desired clustering of lots may differ slightly in cluster sizes: often, large farmers are willing to accept a slight decrease in their total land value, if the redistribution results in a significantly more efficient cultivation.

One of the main challenges is that lots differ in value (aggregated from properties like size, quality of soil, and shape). The restriction on cluster sizes, in combination with the different lot values, makes it provably hard to determine an optimal redistribution of the lots. However, it is possible to compute an approximation of the global optimum based on linear programming over projections of transportation polytopes [4]. Further, the small regions exhibited a heavy repetition of lots of similar values. When there is a fixed bound on the number of different values of lots, for example by rounding lot values to a fixed set of numbers, the underlying clustering problems become tractable [9]. A computed solution in which the lots form large, connected sections of land is depicted in Figure 1b.

However, such a radical redistribution of lots among the farmers (more than 70% of the lots change ownership from Figure 1a to 1b) cannot realistically take place all at once. Crop rotations, required machinery, and other processes for farming stability must be respected.

*University of Colorado Denver (steffen.borgwardt@ucdenver.edu).
†University of Colorado Denver (charles.viss@ucdenver.edu).



(a) Original clustering of lots.

(b) Desired clustering of lots.

Figure 1: Two clusterings of 471 agricultural lots among 7 farmers (images from [3, 4, 5]).

Therefore, the farmers requested a "best" way to gradually implement the proposed changes over the course of several years.

This transition should take the form of a sequence of trades, each changing the ownership on a small subset of lots, and staying between the original and target value of land for all farmers throughout. As before, the differing values of lots make this restriction challenging. However, the heavy repetition of lots of the same (rounded) values can be exploited for two approaches to preprocess the data: first, a restriction to only lots of the same (rounded) values is possible, and then only these lots are traded with each other. Second, instead of trading individual lots, groups of lots of the same farmer that add up to a similar value are traded; to this end, each farmer's lots are partitioned into groups of lots of (or close to) a prescribed total value. With both approaches, a restriction on the value of land of each farmer becomes a restriction on the number of lots, or groups of lots, assigned to the farmer. A good practical performance greatly depends on the repetition of the same lot values.

The result is a representation of the current and target lot distribution as solutions to a new (smaller, or contracted) clustering problem of unweighted items, and a gradual transformation between the clusterings is desired. In this paper, we propose methods for constructing such a transformation based on linear programming and network theory.

The need for the construction of an efficient gradual transition to a new, given clustering also arises in many other applications. For example, an insurance company may want to gradually transition their customers to a new clustering of premium classes. In other situations, there are snapshots of the same data set at different times and the goal is to devise a model that explains how the data gradually changed over time.

In general, we consider partitions of a data set $X := \{x_1, ..., x_n\}$ into k labeled clusters where each item x_i is assigned to exactly one of the clusters. We call such a partition $\mathcal{C} := (C_1, ..., C_k)$ a *k-clustering* of X, or simply a *clustering* of X when k is clear from context. As is the case in most clustering applications, we assume that the number of items is significantly greater than the number of clusters; i.e., $n \gg k$.

We additionally consider situations in which upper and lower bounds are given for the sizes of the clusters. Such bounds may arise directly from an application itself or may be introduced to guide clustering algorithms to return sufficiently balanced solutions. Specifically, let $\kappa^+, \kappa^- \in \mathbb{Z}_+^k$ with $\kappa^+ \geq \kappa^-$ be given. A *bounded-size k-clustering* of X with respect to κ^+ and κ^- then satisfies $\kappa_i^- \leq |C_i| \leq \kappa_i^+$ for $i = 1, ..., k$. This concept generalizes

the *fixed-size k-clusterings* from [2] in which each cluster contains a fixed number of items (i.e., a bounded-size k-clustering with $\kappa^+ = \kappa^-$). The classical transportation problem and the assignment problem, along with their related polytopes, are well-studied topics in optimization corresponding to fixed-size clusterings [1, 14, 22].

For a data set X and cluster size bounds $\kappa^+, \kappa^- \in \mathbb{Z}_+^k$, the *bounded-size partition polytope* $BPP(\kappa^+, \kappa^-)$ models the set of all bounded-size k-clusterings of X with respect to κ^+ and κ^- [12]. Specifically, for $i = 1, ..., k$ and $j = 1, ..., n$, let y_{ij} indicate whether or not cluster C_i receives item x_j in a k-clustering $\mathcal{C} = (C_1, ..., C_k)$ of X. Then $BPP(\kappa^+, \kappa^-)$ is given by the following system of constraints:

$$\begin{aligned}
\sum_{i=1}^{k} y_{ij} &= 1 && j = 1, ..., n \\
\sum_{j=1}^{n} y_{ij} &\geq \kappa_i^- && i = 1, ..., k \\
\sum_{j=1}^{n} y_{ij} &\leq \kappa_i^+ && i = 1, ..., k \\
y_{ij} &\geq 0 && i = 1, ..., k,\ j = 1, ..., n.
\end{aligned}$$

Since these constraints form a totally unimodular matrix, the right-hand side is integral, and each y_{ij} is implicitly bounded between 0 and 1, $BPP(\kappa^+, \kappa^-)$ is in fact a 0/1-polytope whose vertices correspond to the feasible k-clusterings of X. For $\kappa^+ = \kappa^-$, this polytope generalizes the *fixed-size partition polytope* from [2]. It is also an instance of the bounded-shape partition polytope from [7] when X is the standard basis of $\mathbb{R}^n$.

In the fixed-size partition polytope, the edges also have a combinatorial interpretation: two vertices share an edge if and only if the corresponding clusterings differ by a single cyclical move of items among the clusters, formally defined in Section 2. This fact can be used to prove new bounds on the *combinatorial diameter* of the polytope – the maximum length of a shortest edge walk between any pair of vertices – and provide practical methods for constructing transformations between fixed-size clusterings [2]. In this paper, we generalize these methods to the bounded-size partition polytope BPP. Although the edges of this polytope have a more technical characterization, its *circuits* correspond to a set of natural cyclical and sequential moves of items among the clusters [12]. However, constructing transformations using these two types of moves is significantly more challenging than using only cyclical moves due to their different effects on the sizes of the underlying clusters.

Circuits, introduced as the elementary vectors of a subspace by Rockafellar [25], play a fundamental role in the theory of linear programming. For a general polyhedron $P = \{\mathbf{x} \in \mathbb{R}^n \colon A\mathbf{x} = \mathbf{b}, B\mathbf{x} \leq \mathbf{d}\}$, the set of circuits of P consists of all $\mathbf{g} \in \ker(A) \setminus \{\mathbf{0}\}$ normalized to coprime integer components for which $B\mathbf{g}$ is support-minimal over $\{B\mathbf{x} \colon \mathbf{x} \in \ker(A) \setminus \{\mathbf{0}\}\}$. Geometrically, circuits correspond to all potential edge directions of P as the right-hand side vectors $\mathbf{b}$ and $\mathbf{d}$ vary. This implies that the set of circuits serves as a universal test set for any linear program over P [16]. Hence, circuits are used as the step directions in several augmentation algorithms for solving linear programs [11, 13, 17, 18].

Additionally, for polyhedra such as BPP defined by totally unimodular matrices, circuits have combinatorial interpretations in terms of the underlying problem. The support-minimality of circuits implies that, in a sense, steps taken in circuit directions are as simple as possible while maintaining feasibility. In combination with highly-structured problems from combinatorial optimization, these steps become particularly intuitive and are guaranteed to only visit integral points [12].

Therefore, in this paper, we choose the circuits of BPP as the elementary moves for transforming k-clusterings. In building such transformations using these circuits, we construct *circuit walks* between the corresponding vertices in the polytope. As a generalization

of edge walks, circuit walks are of interest due to their relationship to the combinatorial diameter of polyhedra [6, 8, 10]. The *circuit distance* between vertices refers to the minimum number of steps needed to join the vertices by a circuit walk and hence provides a lower bound on the combinatorial distance between the vertices. The related *circuit diameter* of a polyhedron – the minimum number of steps needed to join any pair of vertices by a circuit walk – then serves as a lower bound on its combinatorial diameter. In particular, circuit diameters provide insight into the *polynomial Hirsch Conjecture*, one of the fundamental open questions in linear programming. See [20] for a survey of this field of study.

To build transformations between clusterings, we use one of the main tools from [2] for analyzing fixed-size clusterings. Given two k-clusterings $\mathcal{C}, \mathcal{C}'$ of the same data set, the *clustering-difference graph* $CDG(\mathcal{C}, \mathcal{C}')$ is a directed graph that models the transfers of items required for transforming $\mathcal{C}$ into $\mathcal{C}'$. In the context of fixed-size clusterings, such a graph decomposes into directed cycles. Sets of cyclical moves can then be integrated together in order to bound the combinatorial distance between vertices in the fixed-size partition polytope [2].

We generalize this graph-theoretic approach for constructing transformations between clusterings to the context of general and bounded-size k-clusterings. In these situations, both cycles and paths in a clustering-difference graph correspond to circuits of the related polytopes. Integrating these different types of moves becomes much more technically challenging than integrating only cyclical moves since sequential moves alter the sizes of the underlying clusters. In Section 2, we show how this is possible for certain combinations of cyclical and sequential moves, providing various *double-moves* which reduce the number of steps needed for transforming clusterings (Theorems 2.6 and 2.7). Next, we prove in Section 3 how these double-moves lead to an upper bound (Theorem 3.4) on the so-called *transformation distance* between k-clusterings: a relaxation of the circuit distance. In Section 4 we then prove the implications of this bound on the circuit diameter of the bounded-size partition polytope (Theorem 4.2). We end with a brief discussion on future directions of research in Section 5.

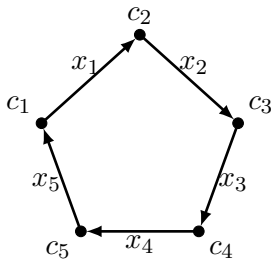
2. Moves and Double-Moves for Transforming Clusterings.

Let $\mathcal{C}, \mathcal{C}'$ be two given k-clusterings of the same data set. We recall from [2] the definition of the *clustering-difference graph* $CDG(\mathcal{C}, \mathcal{C}')$ from $\mathcal{C}$ to $\mathcal{C}'$, a graph-theoretic model for the difference between the two clusterings.

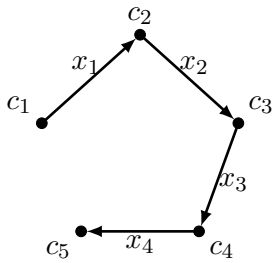
Definition 2.1 (Clustering-difference Graph). *For two k-clusterings $\mathcal{C} = (C_1, ..., C_k)$ and $\mathcal{C}' = (C'_1, ..., C'_k)$ of a data set $X = \{x_1, ..., x_n\}$, the clustering-difference graph $CDG(\mathcal{C}, \mathcal{C}')$ from $\mathcal{C}$ to $\mathcal{C}'$ is a directed arc-labeled multigraph with vertex set $V = \{c_1, ..., c_k\}$ and edge set A, where an edge (c_i, c_j) with label x_ℓ belongs to A if and only if $x_\ell \in C_i$ and $x_\ell \in C'_j$ for $i \neq j$.*

Thus, the edges of $CDG(\mathcal{C}, \mathcal{C}')$ describe all single-item *transfers* needed to transform $\mathcal{C}$ into $\mathcal{C}'$. The number of edges is equal to the number of items whose cluster assignment differs from $\mathcal{C}$ to $\mathcal{C}'$ – if an edge (c_i, c_j) with label x_ℓ belongs to A, item x_ℓ must be moved from cluster i to cluster j as a part of the transformation. Note that this allows for parallel edges in $CDG(\mathcal{C}, \mathcal{C}')$, but all edges have different labels. We refer to a CDG consisting of a single directed cycle – or alternatively to its corresponding set of transfers – as a *cyclical move* of items among the clusters. Similarly, a directed path or its corresponding set of transfers is referred to as a *sequential move* of items. See Figure 2 for examples of these two types of moves.

Throughout the proofs in this paper, we will frequently *construct* cyclical and sequential moves to perform the required clustering transformations. For readability, we introduce the following language to be used when defining such constructions: We say that a move *follows* or *travels along* edges, paths, or cycles of an underlying CDG when we use those parts of the graph to define the transfers of the move, we say that the move uses an *introduced* edge



(a) The clustering-difference graph $CDG(\mathcal{C}, \mathcal{C}')$ when $\mathcal{C}' = (\{x_5\}, \{x_1\}, \{x_2\}, \{x_3\}, \{x_4\})$. The graph is a directed cycle, representing a cyclical move of items among the clusters.



(b) The clustering-difference graph $CDG(\mathcal{C}, \mathcal{C}')$ when $\mathcal{C}' = (\emptyset, \{x_1\}, \{x_2\}, \{x_3\}, \{x_4, x_5\})$. The graph is a directed path, representing a sequential move of items among the clusters.

Figure 2: Given $X = \{x_1, x_2, x_3, x_4, x_5\}$, let $\mathcal{C} = (\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\})$ be a k-clustering of X. For two choices of a different k-clustering $\mathcal{C}'$ of X, Figures 2a and 2b give the corresponding clustering-difference graphs which represent a cyclical and sequential move of items.

when the move contains a transfer that is not reflected by an edge in the underlying CDG, and lastly we say that a move *sends* an item x_ℓ from clustering i to clustering j if one of its transfers is given by the edge (c_i, c_j) with label x_ℓ.

The clustering-difference graph and the associated cyclical and sequential moves play important roles in the analysis of bounded-size and fixed-size partition polytopes. For the fixed-size partition polytope, a CDG decomposes into directed cycles since all vertices in the graph must have equal indegree and outdegree. Two vertices in the polytope then share an edge if and only if the corresponding CDG consists of a single directed cycle. This characterization has been used to construct edge walks between vertices of the polytope using sequences of cyclical moves, resulting in upper bounds on the combinatorial diameter [2].

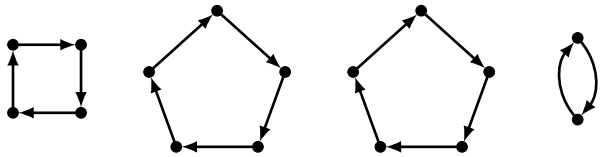
Although the edges of the bounded-size partition polytope have a more complicated characterization, its *circuits* analogously correspond to clustering-difference graphs consisting of either a single directed cycle or a single directed path [12]. Hence, devising sequences of cyclical and sequential moves for transforming $\mathcal{C}$ into $\mathcal{C}'$ can be interpreted as constructing a *circuit walk* between the corresponding vertices in the polytope. As we will show in Sections 3 and 4, this allows us to bound the *circuit distance* between vertices and the related *circuit diameter* of the bounded-size partition polytope.

Therefore, we are interested in constructing a transformation from $\mathcal{C}$ to $\mathcal{C}'$ using as few of these cyclical and sequential moves as possible. We call the minimum number of required moves the *transformation distance* $d(\mathcal{C}, \mathcal{C}')$ from $\mathcal{C}$ to $\mathcal{C}'$.

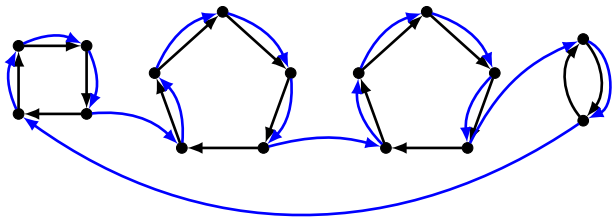
Definition 2.2 (Transformation Distance). *For k-clusterings $\mathcal{C}, \mathcal{C}'$ of the same data set, the transformation distance $d(\mathcal{C}, \mathcal{C}')$ is the minimum number of cyclical and sequential moves needed to transform $\mathcal{C}$ into $\mathcal{C}'$.*

Hence, $d(\mathcal{C}, \mathcal{C}')$ is a relaxation of the circuit distance between the corresponding vertices of the bounded-size partition polytope – as long as no cluster size constraints are violated during a sequence of moves used to achieve $d(\mathcal{C}, \mathcal{C}')$, the two distances are equal.

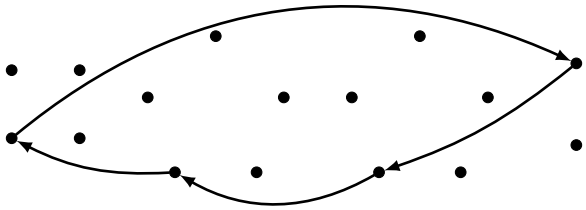
A naive approach for bounding $d(\mathcal{C}, \mathcal{C}')$ is to decompose $CDG(\mathcal{C}, \mathcal{C}')$ into paths and cycles and then simply apply the corresponding moves individually to perform the clustering transformation. However, even when such a decomposition is optimal – i.e., uses as few paths and cycles as possible – $d(\mathcal{C}, \mathcal{C}')$ can be significantly less than the number of parts in the decomposition. For example, Figure 3a depicts a case in which $CDG(\mathcal{C}, \mathcal{C}')$ consists of



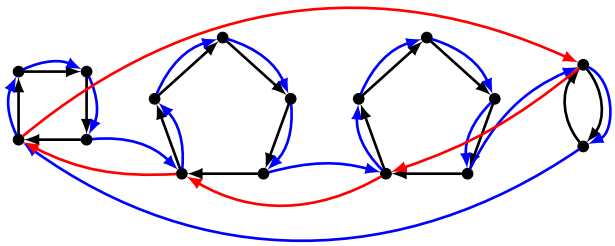
(a) The original clustering-difference graph $CDG(\mathcal{C}, \mathcal{C}')$.



(b) The first cyclical move, given by the blue edges.



(c) The resulting clustering-difference graph after the first cyclical move is applied (and hence, the second cyclical move in the double-move).



(d) A combined visualization of the double-move for transforming $\mathcal{C}$ into $\mathcal{C}'$, with the first move given by the blue edges and the second given by the red edges.

Figure 3: A case where $CDG(\mathcal{C}, \mathcal{C}')$ consists of four disjoint cycles. The above double-move from [2] can be used to transform $\mathcal{C}$ into $\mathcal{C}'$ in only two cyclical moves, implying $d(\mathcal{C}, \mathcal{C}') = 2$.

four vertex-disjoint cycles, which trivially implies $d(\mathcal{C}, \mathcal{C}') \leq 4$. Nevertheless, due to the fact that any permutation can be expressed as the product of two cyclic permutations [1], only two cyclical moves are needed to transform $\mathcal{C}$ into $\mathcal{C}'$, implying $d(\mathcal{C}, \mathcal{C}') = 2$. For simplicity, we will use the term *disjoint* in place of *vertex-disjoint* throughout the remainder of the paper. Whenever two components of a CDG are not vertex-disjoint, we will say that they *intersect*.

Proposition 2.3 ([1], Lemma 7 in [2]). *Let $\mathcal{C}, \mathcal{C}'$ be k-clusterings for which $CDG(\mathcal{C}, \mathcal{C}')$ decomposes into disjoint cycles. Then $d(\mathcal{C}, \mathcal{C}') \leq 2$.*

In [2], Proposition 2.3 is used to integrate disjoint cyclical moves into what we will call a

double-move: a sequence of two moves which results in the desired changes to the underlying clustering-difference graph. See Figure 3 for a visualization of this double-move. In a first cyclical move of items, depicted in Figure 3b, the transfers corresponding to all but one edge from each cycle are correctly applied – the items corresponding to the remaining edges are temporarily sent to incorrect destinations across the cycles. However, a second cyclical move, depicted in Figure 3c, can then be used to send each of these misplaced items to its correct destination, completing the clustering transformation.

In the following lemmas and the upcoming Theorems 2.6 and 2.7, we show how similar double-moves can be used to integrate both cyclical and sequential moves when transforming general k-clusterings. This becomes more technically challenging due to the different natures of the two types of moves – cyclical moves do not alter the sizes of any underlying cluster while sequential moves alter the sizes of exactly two clusters by one. First, generalizing Proposition 2.3, we show how to integrate sets of disjoint cycles and paths from a CDG. Note that the bound on $d(\mathcal{C}, \mathcal{C}')$ in the following lemma depends only on the number of paths in the decomposition and not on the number of cycles.

Lemma 2.4. *Let $\mathcal{C}, \mathcal{C}'$ be k-clusterings for which $CDG(\mathcal{C}, \mathcal{C}')$ decomposes into disjoint cycles and paths. Then $d(\mathcal{C}, \mathcal{C}') \leq \max\{2,\ t\}$, where t is the number of paths in the decomposition.*

Proof. Let t denote the number of directed paths and s the number of directed cycles in the decomposition of $CDG(\mathcal{C}, \mathcal{C}')$. When $t = 0$, the result follows from Proposition 2.3, so we assume $t \geq 1$. We also assume $s \geq 1$, else the t sequential moves could simply be applied individually.

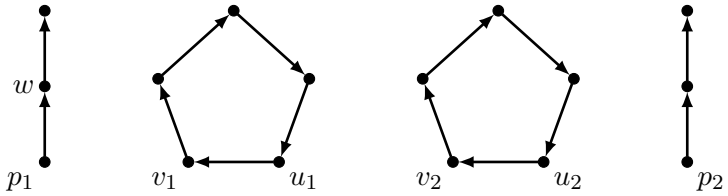
First suppose $t = 2$. Let P_1, P_2 denote the paths of $CDG(\mathcal{C}, \mathcal{C}')$ and let $Y_1, ..., Y_s$ denote the cycles. For $i = 1, ..., s$, we select any edge $e_i := (u_i, v_i)$ from cycle Y_i. In addition, let p_1 denote the tail of P_1, p_2 the tail of P_2, and w the neighbor of p_1 on P_1. See Figure 4a. We will apply a double-move consisting of two sequential moves to transform clustering $\mathcal{C}$ into $\mathcal{C}'$.

The first sequential move is constructed as follows: First, an edge is introduced from p_1 to v_1, sending to v_1 the item from p_1 intended for w. After traveling along this edge, the move follows $Y_1 - e_1$ from v_1 to u_1. Next, if $s > 1$, it uses an introduced edge from u_1 to v_2 whose label is that of e_1, and then it travels along $Y_2 - e_2$ to u_2. This is repeated for $i = 2, ..., s - 1$ until u_s is reached. The move terminates by using an introduced edge from u_s to p_2 whose label is that of e_s, and then by traveling along path P_2. See the blue edges in Figure 4b for a visualization of this move.

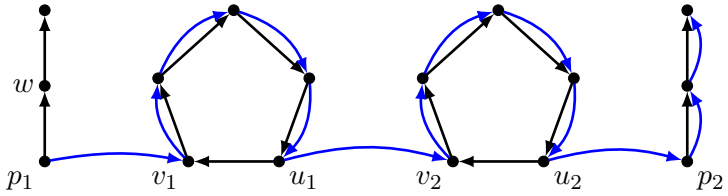
Note that this first sequential move applies all transfers given by the cycles $Y_1, ..., Y_s$ in $CDG(\mathcal{C}, \mathcal{C}')$ with the exception of those corresponding to the edges e_i – the item from u_i intended for v_i is temporarily sent to the wrong cluster. The move also applies all transfers given by P_2 but does not correctly apply any transfer from P_1. However, all misplaced items can be corrected and all remaining transfers can be applied via a single, second sequential move as seen in Figure 4c. This sequential move first sends from p_2 to v_s the item p_2 received from u_s. Next, for $i = s, ..., 2$, it sends from v_i to v_{i-1} the item v_i received from u_{i-1}. Once v_1 is reached, it sends from v_1 to w the item v_1 received from p_1. The move terminates by following the remaining edges of P_1. After this second sequential move is applied, the transformation from $\mathcal{C}$ to $\mathcal{C}'$ is complete – it follows that $d(\mathcal{C}, \mathcal{C}') = 2$. See Figure 4d for a visualization of these moves integrated together into a double-move

Next suppose $t > 2$. We can apply the double-move from the previous case to remove all cycles and any two paths from $CDG(\mathcal{C}, \mathcal{C}')$. The remaining $t - 2$ paths can then be applied via individual sequential moves. Hence, $d(\mathcal{C}, \mathcal{C}') \leq t$.

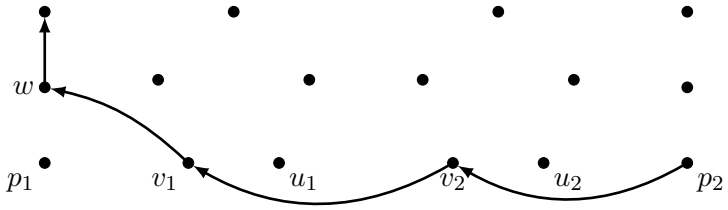
Lastly, suppose $t = 1$. Again let P_1 denote the single path of $CDG(\mathcal{C}, \mathcal{C}')$, let p_1 denote the tail of P_1, and let w denote the neighbor of p_1 on P_1. We choose an edge $e_i := (u_i, v_i)$ from Y_i for $i = 1, ..., s$ and apply a sequential move followed by a cyclical move to transform $\mathcal{C}$ into $\mathcal{C}'$. See Figure 5 for a visualization of the double-move.



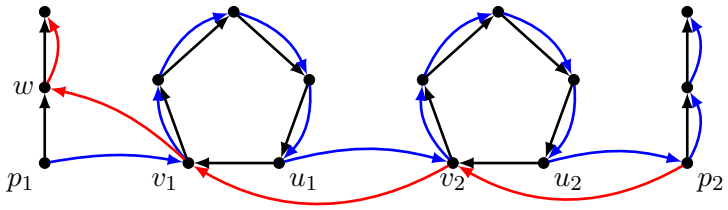
(a) The original clustering-difference graph $CDG(\mathcal{C}, \mathcal{C}')$.



(b) The first sequential move.



(c) The resulting CDG after the first sequential move is applied. This directed path corresponds to the second sequential move used to transform $\mathcal{C}$ into $\mathcal{C}'$.



(d) A combined visualization of the double-move for transforming $\mathcal{C}$ into $\mathcal{C}'$, with the first move given by the blue edges and the second given by the red edges.

Figure 4: A case where $CDG(\mathcal{C}, \mathcal{C}')$ consists of two paths and multiple cycles. Independent of the number of cycles, only two sequential moves are needed to transform $\mathcal{C}$ into $\mathcal{C}'$, so $d(\mathcal{C}, \mathcal{C}') = 2$.

As in the previous case, the sequential move first uses an introduced edge from p_1 to v_1 and then travels along $Y_1 - e_1$ from v_1 to u_1. If $s > 1$, then for $i = 1, ..., s-1$, it uses an introduced edge from u_i to v_i and follows $Y_i - e_i$ to u_i until u_s is reached. The move terminates by using an introduced edge from u_s to w, and then by following the remaining edges of P_1.

The second cyclical move corrects all items sent to incorrect destinations by the first move. Namely, it first follows the edge from w to v_s, sending to v_s the item w received from u_s. Next, for $i = s, ..., 2$, it follows the edges from v_i to v_{i-1} until v_1 is reached. The move terminates by following the edge from v_1 to w, sending w the item v_1 received from p_1. This completes the transformation from $\mathcal{C}$ to $\mathcal{C}'$, so $d(\mathcal{C}, \mathcal{C}') \leq 2$.

Therefore, in each case, $d(\mathcal{C}, \mathcal{C}') \leq \max\{2,\ t\}$. □

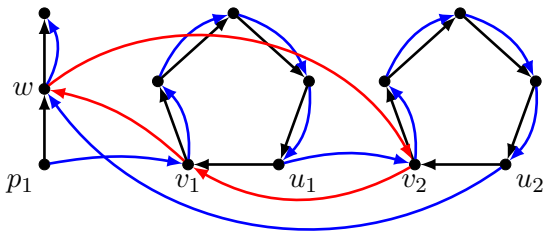


Figure 5: A visualization of the double-move for transforming $\mathcal{C}$ into $\mathcal{C}'$ when $CDG(\mathcal{C}, \mathcal{C}')$ consists of a single path and multiple cycles. The original $CDG(\mathcal{C}, \mathcal{C}')$ is given by the black edges, the first sequential move is given by the blue edges, and the second cyclical move is given by the red edges.

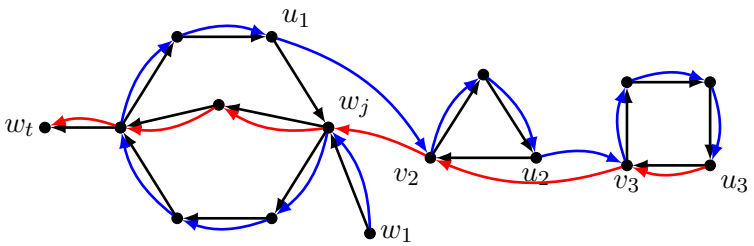


Figure 6: A visualization of the double-move for transforming $\mathcal{C}$ into $\mathcal{C}'$ when $CDG(\mathcal{C}, \mathcal{C}')$ consists of multiple disjoint cycles and a path intersecting a single cycle.

Even when paths and cycles are not completely disjoint, the corresponding moves may still be integrated together into a double-move. In the following lemma, we show that if a path intersects at most one cycle out of a collection of disjoint cycles, only two moves are required to apply all corresponding transfers.

Lemma 2.5. *Let $\mathcal{C}, \mathcal{C}'$ be k-clusterings for which $CDG(\mathcal{C}, \mathcal{C}')$ consists of $s \geq 1$ disjoint cycles and a single path P, which intersects at most one of the cycles. Then $d(\mathcal{C}, \mathcal{C}') \leq 2$.*

Proof. Let $Y_1, ..., Y_s$ denote the cycles of $CDG(\mathcal{C}, \mathcal{C}')$. If P does not intersect any of the cycles, we can apply Lemma 2.4, so we assume P intersects Y_1. We also assume $s \geq 2$, else the sequential and cyclical moves corresponding to P and Y_1 could simply be applied individually. Let $w_1, ..., w_t$ denote (in order) the vertices of P. We select an edge $e_i := (u_i, v_i)$ from each cycle Y_i, where e_1 is chosen such that v_1 is the first vertex w_j of P that also belongs to Y_1. (Note that it is possible to have $j = 1$ or $j = t$, but both of those cases are still covered by the following construction.) We will apply two sequential moves to transform $\mathcal{C}$ into $\mathcal{C}'$. See Figure 6 for a visualization of $CGD(\mathcal{C}, \mathcal{C}')$ along with the double-move.

The first sequential move starts at w_1 and follows P to $w_j = v_1$. It then follows the path formed by joining $Y_1 - e_1, ..., Y_s - e_s$ with the introduced edges $(u_1, v_2), ..., (u_{s-1}, v_s)$, terminating at u_s. Hence, the move reduces the size of the cluster corresponding to w_1, as desired, but it also increases the size of the cluster corresponding to u_s.

To correct this, we apply a second sequential move starting at u_s and terminating at w_t. The move first follows the edges $(u_s, v_s), (v_s, v_{s-1}), ..., (v_2, v_1)$ to correct items misplaced across cycles. Next, since $v_1 = w_j$, it terminates by following P along the vertices $w_j, ..., w_t$. Since P only intersects Y_1, no vertices are repeated in this second sequential move and it indeed corresponds to a single directed path. All transfers corresponding to the original edges of $CDG(\mathcal{C}, \mathcal{C}')$ have then been correctly applied. □

(a) The clustering-difference graph $CDG(\mathcal{C}, \mathcal{C}')$.

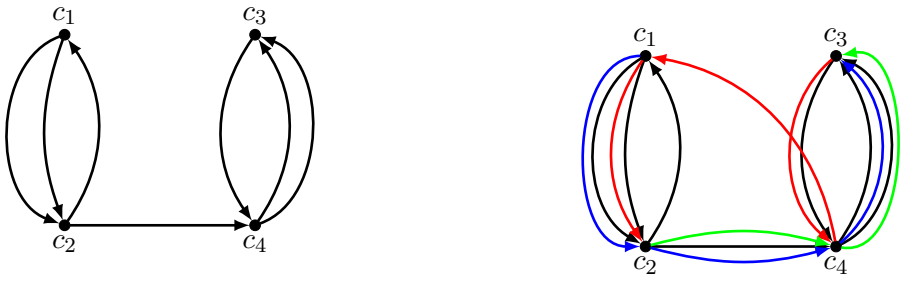
(b) The only double-move (apart from symmetry) that can be used to transform $\mathcal{C}$ into $\mathcal{C}'$.

Figure 7: A clustering-difference graph $CDG(\mathcal{C}, \mathcal{C}')$ for which the only way to transform $\mathcal{C}$ into $\mathcal{C}'$ in only two moves is to use the double-move from Lemma 2.5.

Note that in the double-move used for Lemma 2.5, the first sequential move temporarily increases the size of the cluster corresponding to u_s. For general k-clusterings this is not an issue, but for bounded-size k-clusterings this could potentially violate the upper bound on the size of the cluster. Unfortunately, this increase in cluster size is unavoidable for certain configurations of $CDG(\mathcal{C}, \mathcal{C}')$. Consider the example in Figure 7. If only two moves are to be used to transform $\mathcal{C}$ into $\mathcal{C}'$, c_1 must correctly send an item to c_2 in each move. Hence, c_1 may not temporarily receive an item from c_3 or c_4 in the first move, and the cluster size corresponding to either c_3 or c_4 must be temporarily increased. Therefore, although $d(\mathcal{C}, \mathcal{C}') = 2$, the circuit distance between the corresponding vertices in a related bounded-size partition polytope may be 3 if the cluster sizes corresponding to c_3 and c_4 are already at their upper bounds. We further address this issue in Section 4.

When a path intersects multiple cycles from a set of disjoint cycles, integrating the corresponding moves as in the double-move from Lemma 2.5 becomes more challenging since we can no longer guarantee that the second part of the move corresponds to a single directed path in the underlying CDG. See for example Figure 8, in which all four vertices have either indegree or outdegree at least two. In such a scenario, each cluster must either correctly send or correctly receive two items to perform the clustering transformation, implying that no item can be temporarily sent to an incorrect destination during a double-move. Thus at least three moves are required. However, we can ensure that transfers corresponding to at least the *first* and *last* edges of the path are applied in conjunction with the disjoint cycles. We show in the upcoming theorem that given any path $P = w_1 ... w_t$ and a set $\mathcal{Y}$ of disjoint cycles, a double-move can be used to correctly apply all transfers from $\mathcal{Y}$ while decreasing the cluster size corresponding to w_1 and increasing the cluster size corresponding to w_t. Furthermore, such a double-move will then allow us to completely integrate P with $\mathcal{Y}$ as long as P does not intersect the cycles of $\mathcal{Y}$ more than three times.

Recall that in a clustering-difference graph $CDG(\mathcal{C}, \mathcal{C}')$, the outdegree of a vertex is equal to the number of items which must be moved from the corresponding cluster to perform the clustering transformation. Note that in order for the outdegree to be reduced, a correct item must be sent from the cluster to a new destination; however, this destination cluster need not actually be the other endpoint of the corresponding edge in $CDG(\mathcal{C}, \mathcal{C}')$. On the other hand, the indegree of a vertex gives the number of items which must be moved to the corresponding cluster. For the indegree to be reduced, a correct item must be *received* by the vertex, but it does not matter which cluster actually sends the item. We call the minimum of the indegree and outdegree of a vertex the *shared degree* of that vertex in the CDG. Hence, applying a set of disjoint cyclical moves reduces the shared degree of all covered vertices by one. When integrating a path with these cyclical moves, this reduction in shared degree should still occur in order to make all desired improvements



(a) The clustering-difference graph $CDG(\mathcal{C}, \mathcal{C}')$.

(b) A triple-move for transforming $\mathcal{C}$ into $\mathcal{C}'$.

Figure 8: A clustering-difference graph $CDG(\mathcal{C}, \mathcal{C}')$ with a path intersecting multiple disjoint cycles such that three moves are required to perform the corresponding clustering transformation.

to the underlying CDG. In the following theorem, we provide four different double-moves to accomplish this task. The type of double-move to use depends on the intersection points of the path with the cycles.

Theorem 2.6. *Let $\mathcal{C}, \mathcal{C}'$ be k-clusterings of the same data set with clustering-difference graph $D := CDG(\mathcal{C}, \mathcal{C}')$, let $\mathcal{Y} = Y_1, ..., Y_s$ be a set of disjoint cycles in D, and let $P = w_1 ... w_t$ be a path in D that is edge-disjoint from $\mathcal{Y}$. There exists a double-move which accomplishes all of the following:*

1. *Correctly applies all transfers from $\mathcal{Y}$*
2. *Reduces the cluster size corresponding to w_1 through sending a correct item*
3. *Increases the cluster size corresponding to w_t through receiving a correct item*
4. *Decreases the shared degree of each vertex covered by $\mathcal{Y}$ by at least one.*

Proof. If $s = 1$ or if P is disjoint from $\mathcal{Y}$, we can apply Lemma 2.5, so we assume $s \geq 2$. Let w_{j_1} denote the first vertex of P covered by $\mathcal{Y}$, and let w_{j_2}, w_{j_3} denote, respectively, the second-to-last and last vertex of P covered by $\mathcal{Y}$. Note that we have $w_{j_1} = w_{j_2}$ when P intersects $\mathcal{Y}$ only twice. Similarly, if P intersects $\mathcal{Y}$ only once, we let $w_{j_1} = w_{j_2} = w_{j_3}$. We treat four exhaustive cases regarding the distribution of w_{j_1}, w_{j_2}, and w_{j_3} across the cycles of $\mathcal{Y}$.

Case 1: w_{j_1} and w_{j_3} belong to different cycles of $\mathcal{Y}$ and w_{j_2} belongs to the same cycle as w_{j_3}. We apply a cyclical move followed by a sequential move to perform all necessary transfers in D. See Figure 9 for a visualization of the double-move.

Without loss of generality, we assume $w_{j_1} \in Y_1$ and $w_{j_2}, w_{j_3} \in Y_s$. We choose an edge $e_i := (u_i, v_i)$ from each cycle Y_i, where e_1 and e_s are chosen such that $u_1 = w_{j_1}$ and $v_s = w_{j_2}$. The cyclical move first introduces an edge from w_{j_1} to $w_{j_2} = v_s$ whose label is the item sent from w_{j_1} in P. (Note that we might have $w_{j_1} = w_{j_2 - 1}$, in which case the edge already exists in $CDG(\mathcal{C}, \mathcal{C}')$.) Next, the move follows Y_s from v_s to u_s. It then uses an introduced edge from u_s to v_{s-1} whose label is that of e_s, and travels along $Y_{s-1} - e_{s-1}$ until u_{s-1} is reached. This is repeated for the remaining cycles via introduced edges (u_i, v_{i-1}) and by traveling along $Y_{i-1} - e_{i-1}$ until $u_1 = w_{j_1}$ is reached, completing the cyclical move.

Next, the sequential move starts at w_1, following P to $w_{j_1} = u_1$, and then follows the edge (u_1, v_1). It then follows the edges (v_i, v_{i+1}) for $i = 1, ..., s-1$ until $v_s = w_{j_2}$ is reached, correcting items misplaced across the cycles. The move terminates by following P from w_{j_2} to w_{j_3} and then to w_t.

We now prove that the desired changes have been made to the underlying CDG. Clearly the cluster size corresponding to w_1 is decreased and the cluster size corresponding to w_t is increased via the second sequential move. Furthermore, each item sent by w_1 or received by

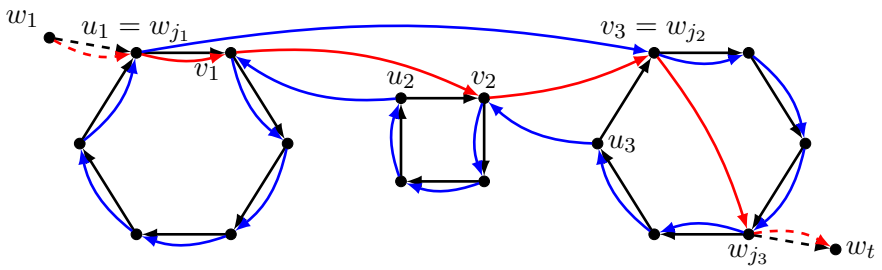


Figure 9: The double-move for Case 1 of Theorem 2.6. The cycles of $\mathcal{Y}$ are given by the solid black edges, the first cyclical move is given by the blue edges, and the second sequential move is given by the red edges. The dashed edges indicate the sections of P that exist when $w_1 \neq w_{j_1}$ or $w_t \neq w_{j_3}$. Note that when $w_{j_2} \neq w_{j_3-1}$, the second sequential move will include additional vertices of P between w_{j_2} and w_{j_3} which are not covered by $\mathcal{Y}$.

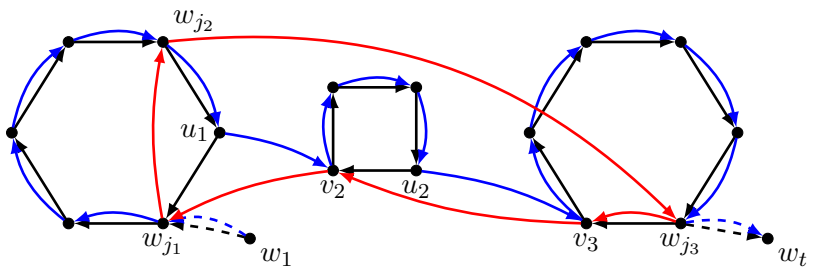


Figure 10: The double-move for Case 2 of Theorem 2.6.

w_t is correct. As in the double-move from Figure 3, all edges from $\mathcal{Y}$ are applied through the combination of the two moves. Thus, it suffices to show that the shared degree of each vertex covered by $\mathcal{Y}$ has been reduced by at least one. The only interesting cases are w_{j_1}, w_{j_2}, and w_{j_3}. In the first cyclical move, w_{j_1} both receives and sends a correct item, reducing its shared degree by one. In the following sequential move, either w_{j_1} only sends a correct item (if $w_{j_1} = w_1$) or w_{j_1} both sends and receives a correct item, so the net reduction in shared degree of w_{j_1} is at least one. Similarly, w_{j_3} sends and receives a correct item in the first move and then either receives a correct item or both receives and sends a correct item in the second move.

Finally, consider w_{j_2}. The vertex receives a possibly incorrect item from w_{j_1} but also sends a correct item to its neighbor on Y_s via the first cyclical move, leaving its shared degree, at worst, unchanged. In the second sequential move, w_{j_2} receives a correct item originating from u_s and then sends a correct item to the following vertex on P. Thus, the shared degree of w_{j_2} is also reduced by at least one, as desired.

Case 2: w_{j_1} and w_{j_3} belong to different cycles of $\mathcal{Y}$ and w_{j_2} does not belong to the same cycle as w_{j_3}. (w_{j_2} may or may not belong to the same cycle as w_{j_1}, or we may have $w_{j_1} = w_{j_2}$.) We apply a sequential move followed by a cyclical move to perform the necessary transfers. See Figure 10 for a visualization of the double-move.

We assume without loss of generality that $w_{j_1} \in Y_1$ and $w_{j_3} \in Y_s$, and then choose an edge $e_i := (u_i, v_i)$ from each Y_i such that $v_1 = w_{j_1}$, $u_s = w_{j_3}$, and $v_i \neq w_{j_2}$ for $i = 2, ..., s-1$. First, the sequential move travels along P from w_1 to w_{j_1}. Then for $i = 1, ..., s-1$, it travels along $Y_i - e_i$ and follows an introduced edge (u_i, v_{i+1}). The move terminates by following $Y_s - e_s$ from v_s to $u_s = w_{j_3}$ and then following P from w_{j_3} to w_t.

The cyclical move starts at $w_{j_3} = u_s$ by following the edge (u_s, v_s). Then for $i = s, ...2$, it follows the edges (v_i, v_{i-1}) until $v_1 = w_{j_1}$ is reached, correcting items misplaced across

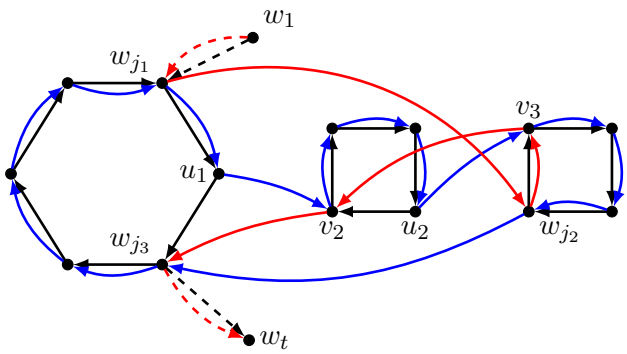


Figure 11: The double-move for Case 3 of Theorem 2.6.

the cycles. Next, the move follows an introduced (or possibly existing) edge from w_{j_1} to w_{j_2} whose label is the item sent from w_{j_1} in P. It then terminates by following P from w_{j_2} to w_{j_3}. Note that this is indeed a single cyclical move since $w_{j_2} \neq v_i$ for $i = 2, ..., s$.

The first sequential move alters the cluster sizes corresponding to w_1 and w_t through correct transfers as desired, and again the edges of $\mathcal{Y}$ are all applied through the combination of the two moves. It suffices to show that the double-move reduces the shared degree of all vertices covered by $\mathcal{Y}$. However, this again follows from the argument used in the previous case: although w_{j_2} may receive an incorrect item from w_{j_1}, it sends away two correct items and its shared degree is reduced by at least one.

Case 3: w_{j_1} and w_{j_3} belong to the same cycle of $\mathcal{Y}$ but w_{j_2} belongs to a different cycle. We assume $w_{j_1}, w_{j_3} \in Y_1$ and $w_{j_2} \in Y_s$, and we apply a cyclical move followed by a sequential move to perform the necessary transfers. See Figure 11 for a visualization of the double-move.

We choose an edge $e_i := (u_i, v_i)$ from each cycle Y_i such that $v_1 = w_{j_3}$ and $u_s = w_{j_2}$. The cyclical move first travels along $Y_1 - e_1$ from $v_1 = w_{j_3}$ to u_1. Then for $i = 1, ..., s-1$, it uses an introduced edge (u_i, v_{i+1}) and follows $Y_{i+1} - e_{i+1}$ to u_{i+1}. Once $u_s = w_{j_2}$ is reached, the move terminates by following P from w_{j_2} to w_{j_3}.

The sequential move starts by following P from w_1 to w_{j_1}. Next, it follows an introduced (or possibly existing) edge from w_{j_1} to $w_{j_2} = u_s$ whose label is the item sent from w_{j_1} in P, and then follows the edge (u_s, v_s). Next, for $i = s, ..., 2$, it follows the edge (v_i, v_{i-1}) to correct items misplaced among the cycles. Once $v_1 = w_{j_3}$ is reached, the move terminates by following P from w_{j_3} to w_t. All clusters are then changed as desired using the arguments from Case 1.

Case 4: w_{j_1}, w_{j_2}, and w_{j_3} belong to the same cycle in $\mathcal{Y}$ (allowing for either $w_{j_1} = w_{j_2}$ or $w_{j_1} = w_{j_2} = w_{j_3}$). We apply two sequential moves similar to those of Lemma 2.5 to perform all necessary transfers. See Figure 12 for a visualization of the double-move.

We assume without loss of generality that $w_{j_1}, w_{j_2}, w_{j_3} \in Y_1$, and we choose an edge $e_i := (u_i, v_i)$ from each cycle Y_i, where e_1 is chosen such that $v_1 = w_{j_1}$. The first sequential move starts by following P from w_1 to $w_{j_1} = v_1$ and then travels along $Y_1 - e_1$ to u_1. Then for $i = 1, ..., s-1$, it uses an introduced edge (u_i, v_{i+1}) and travels along $Y_i - e_i$. It terminates once u_s is reached.

The second sequential move begins at u_s by following the edge e_s. Then for $i = s, ..., 2$, it travels along the edge (v_i, v_{i-1}) to correct items misplaced among the cycles. Once $v_1 = w_{j_1}$ is reached, the move follows an introduced (or possibly existing) edge from w_{j_1} to w_{j_2} whose label is the item sent from w_{j_1} in P. It then terminates by following P from w_{j_2} to w_{j_3} and then to w_t. All clusters are then changed as desired by again using the arguments from Case 1. □

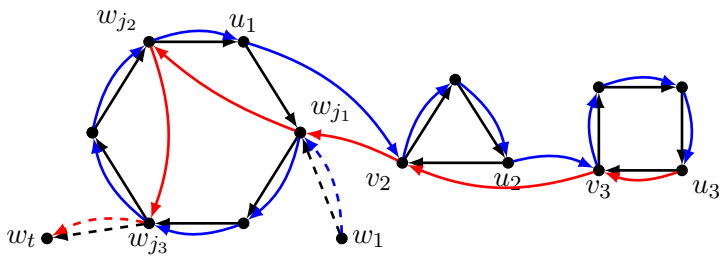


Figure 12: The double-move for Case 4 of Theorem 2.6.

In the proof of Theorem 2.6, we observe an important implication: if a path intersects a set of disjoint cycles at most three times, then *all* transfers corresponding to the cycles and the path can be correctly applied using one of the double-moves from the theorem.

Theorem 2.7. *Let $\mathcal{C}, \mathcal{C}'$ be k-clusterings where $CDG(\mathcal{C}, \mathcal{C}')$ consists of a set $\mathcal{Y}$ of disjoint cycles and a path P. If P intersects $\mathcal{Y}$ at most three times, then $d(\mathcal{C}, \mathcal{C}') = 2$.*

Proof. Let $P = w_1 ... w_t$. As in the proof of Theorem 2.6, consider w_{j_1}, w_{j_2}, and w_{j_3}: the first, second-to-last, and last vertices of P which are covered by $\mathcal{Y}$. If these are the only vertices of P covered by $\mathcal{Y}$, then all edges of P can be applied in any of the four double-moves from Theorem 2.6. To see this, note that since no vertices on P between w_{j_1} and w_{j_2} are covered by $\mathcal{Y}$, we can follow *all* corresponding edges of P when sending an item from w_{j_1} to w_{j_2}. As in the proof of the theorem, the same holds for all edges between w_{j_2} and w_{j_3}. Hence, all transfers corresponding to the edges from both $\mathcal{Y}$ and P are correctly applied through the appropriate double-move, implying $d(\mathcal{C}, \mathcal{C}') \leq 2$. Note that since $CDG(\mathcal{C}, \mathcal{C}')$ either contains disjoint components or vertices with shared degree greater than one, $d(\mathcal{C}, \mathcal{C}') > 1$. Therefore, $d(\mathcal{C}, \mathcal{C}') = 2$. □

Of course, since the number of times a path P intersects a set $\mathcal{Y}$ of cycles is at most the number of vertices in P, this implies that a path with at most three vertices can always be integrated with $\mathcal{Y}$ using one of the double-moves from Theorem 2.6.

Corollary 2.8. *Let $\mathcal{C}, \mathcal{C}'$ be k-clusterings where $CDG(\mathcal{C}, \mathcal{C}')$ consists of a set $\mathcal{Y}$ of disjoint cycles and a path P with at most three vertices. Then $d(\mathcal{C}, \mathcal{C}') = 2$.*

3. Bounds on the Transformation Distance.

In this section, we use the double-moves from Section 2 to prove upper bounds on the transformation distance between clusterings based on certain properties of the related CDG.

Given any two k-clusterings $\mathcal{C}, \mathcal{C}'$ with clustering-difference graph $D := CDG(\mathcal{C}, \mathcal{C}')$, our goal is to transform $\mathcal{C}$ into $\mathcal{C}'$ using as few cyclical and sequential moves as possible. Recall the fundamental difference between these two types of moves: cyclical moves transfer items among the clusters while preserving the original cluster sizes; on the other hand, sequential moves transfer items while increasing the size of one cluster and decreasing the size of another. This motivates a decomposition of D into two parts corresponding to these different types of moves.

Definition 3.1 (Path-Cycle Decomposition). *Let $\mathcal{C}, \mathcal{C}'$ be k-clusterings of the same data set with clustering-difference graph $D := CDG(\mathcal{C}, \mathcal{C}')$. For $i = 1, ..., k$, let δ_i denote $|(|C_i| - |C_i'|)|$, the change in the size of cluster C_i between $\mathcal{C}$ and $\mathcal{C}'$. A path-cycle decomposition (D_P, D_Y) of D is a decomposition of D into two parts: a set D_P containing $\frac{1}{2}\sum_{i=1}^{k} \delta_i$ directed paths and a graph D_Y which decomposes into directed cycles.*

For any path-cycle decomposition (D_P, D_Y) of D, the paths of D_P adjust the cluster sizes of $\mathcal{C}$ to those of $\mathcal{C}'$ and the edges of D_Y apply any remaining transfers. Such a decomposition can be found easily: one greedily constructs directed paths in D which begin

at *excess vertices*, those vertices c_i with outdegree greater than indegree (i.e., $d^+(c_i) > d^-(c_i)$), and which terminate at *deficit vertices*, those satisfying $d^-(c_i) > d^+(c_i)$. These paths are added to D_P. Once there do not exist any excess or deficit vertices in D, the remaining edges in the graph form D_Y. Alternatively, one can greedily remove directed cycles from D to build D_Y and the leftover edges will decompose into D_P. Note that we can store D_P either as a set of directed paths or as a graph which decomposes into paths. Nevertheless, the fixed number of paths from excess to deficit vertices in D_P gives a lower bound on the transformation distance between the clusterings.

Lemma 3.2. *Let $\mathcal{C}, \mathcal{C}'$ be k-clusterings of the same data set. Then $d(\mathcal{C}, \mathcal{C}') \geq \frac{1}{2}\sum_{i=1}^{k} \delta_i$, where $\delta_i = |(|C_i| - |C_i'|)|$.*

Proof. By definition, δ_i is the change in cluster size needed to transform C_i into C_i'. Note that the sum $\sum_{i=1}^{k} \delta_i$ is therefore even. Cyclical moves do not change the size of any clusters while sequential moves change the size of exactly two clusters by one. Hence, at least $\frac{1}{2}\sum_{i=1}^{k} \delta_i$ sequential moves are required in order to change the cluster sizes of $\mathcal{C}$ to those of $\mathcal{C}'$. □

Given a path-cycle decomposition (D_P, D_Y) of D, a straightforward approach for transforming $\mathcal{C}$ into $\mathcal{C}'$ is to separately apply the paths of D_P followed by the cycles of D_Y. However, whereas a fixed number of sequential moves is required to apply all paths of D_P, the number of cyclical moves required to apply all transfers in D_Y is generally less than its number of cycles. Using the double-move from Figure 3, we can integrate sets of disjoint cycles from D_Y to achieve a transformation distance bound which generalizes Corollary 7 in [2]. This serves as a starting point for our discussion on an improved upper bound for $d(\mathcal{C}, \mathcal{C}')$. Recall that the *shared degree* η_i of a vertex c_i in D is the minimum of its indegree and outdegree.

Lemma 3.3. *Let $\mathcal{C}, \mathcal{C}'$ be k-clusterings of the same data set. Then*

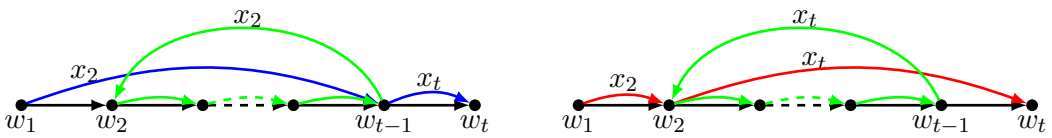
$$d(\mathcal{C}, \mathcal{C}') \leq \eta_{i_1} + \eta_{i_2} + \frac{1}{2}\sum_{i=1}^{k} \delta_i,$$

where $\delta_i = |(|C_i| - |C_i'|)|$, η_i is the shared degree of c_i in $CDG(\mathcal{C}, \mathcal{C}')$, $i_1 = \arg\max \eta_i$, and $i_2 = \arg\max_{i \neq i_1} \eta_i$.

Proof. Let (D_P, D_Y) be any path-cycle decomposition of $D := CDG(\mathcal{C}, \mathcal{C}')$. Applying the $\frac{1}{2}\sum_{i=1}^{k} \delta_i$ sequential moves given by the paths of D_P correctly adjusts all cluster sizes.

Next, we can use the method of Corollary 7 from [2] to apply the cycles in D_Y. To do so, note that for $i = 1, ..., k$, the shared degree of c_i in D_Y is at most η_i. Hence, we may first apply at most $\eta_{i_1} - \eta_{i_2}$ cyclical moves to reduce the maximum shared degree in D_Y to at most η_{i_2}. Next, using the technique in Corollary 3 from [2], we can solve a maximum flow problem to obtain a set of disjoint cycles in D_Y covering all vertices of maximum shared degree. All transfers from this cycle cover can be applied via at most two cyclical moves using the double-move from Figure 3. Repeating until the maximum shared degree of D_Y is zero, all transfers from D_Y are performed in at most $(\eta_{i_1} - \eta_{i_2}) + 2\eta_{i_2} = \eta_{i_1} + \eta_{i_2}$ cyclical moves. Therefore, at most $\eta_{i_1} + \eta_{i_2} + \frac{1}{2}\sum_{i=1}^{k} \delta_i$ cyclical and sequential moves are required to transform $\mathcal{C}$ into $\mathcal{C}'$. □

This initial upper bound on $d(\mathcal{C}, \mathcal{C}')$ uses the double-move for integrating disjoint cyclical moves from Figure 3 but does not yet take advantage of any of the double-moves from Section 2 which integrate both cyclical and sequential moves. For instance, when applying the cyclical moves from a disjoint cycle cover $\mathcal{Y}$ of all vertices of maximum shared degree in D_Y, we could attempt to integrate a path P from D_P. If P is disjoint from $\mathcal{Y}$, if P intersects at most one cycle of $\mathcal{Y}$, or if P intersects $\mathcal{Y}$ at most three times, we could use one of the double-moves from Lemma 2.4, Lemma 2.5, or Theorem 2.7 to integrate P at



(a) The first case for P' and e_P.

(b) The second case for P' and e_P.

Figure 13: The sequential move $P = w_1...w_t$, given by the black edges, is equivalent to one of two combinations of a short sequential move with a cyclical move. In the first case, the path $P' := w_1 w_{t-1} w_t$, given by the blue edges, is applied first. Next, the cycle P_Y, given by the green edges, is applied and sends x_2 to the correct destination w_2 via the introduced edge $e_P := (w_{t-1}, w_2)$. In the second case, the cycle P_Y is applied first, sending x_t from w_{t-1} to w_2 via e_P. Then the path $P' := w_1 w_2 w_t$, given by the red edges, is applied and x_t is correctly sent from w_2 to w_t. In both cases, all transfers corresponding to the original path P are performed correctly via the path P' and the cycle P_Y.

no extra cost, reducing the number of remaining sequential moves. However, we cannot guarantee that such a path P exists in D_P.

Nevertheless, we can achieve an improved bound on the transformation distance by considering each path in D_P as the combination of a (short) sequential move with a cyclical move. To motivate this, suppose $CDG(\mathcal{C}, \mathcal{C}')$ consists of a set $\mathcal{Y}$ of disjoint cycles and a path $P = w_1...w_t$ with $t \geq 4$, where w_1, w_{t-1}, and w_t are all covered by $\mathcal{Y}$. We can apply one of the four double-moves from Theorem 2.6 to perform all transfers corresponding to the edges in $\mathcal{Y}$ while decreasing the excess of w_1 and the deficit of w_t. However, the double-move does not apply any edges of P between w_2 and w_{t-1}. Additionally, w_{t-1} receives an incorrect item from w_1 during the double-move which still needs to be sent to w_2. Hence, a new edge from w_{t-1} to w_2 is introduced and the resulting CDG consists of the directed cycle $w_2...w_{t-1}w_2$.

In this manner, we can represent the sequential move corresponding to any path $P = w_1...w_t$ in D_P covering $t \geq 4$ vertices as the combination of such a cycle $P_Y = w_2...w_{t-1}w_2$ and a path P' with three vertices. As depicted in Figure 13, there are two cases to consider regarding the interior vertex of P' and the item sent along the artificially introduced edge $e_P := (w_{t-1}, w_2)$ in P_Y. These cases depend on the order in which the corresponding transfers are applied. Let x_2 denote the item to be sent from w_1 to w_2 in P, and let x_t denote the item to be sent from w_{t-1} to w_t. If P' is applied first (as in the previous paragraph), let $P' := w_1 w_{t-1} w_t$ and let e_P send x_2 from w_{t-1} to the correct destination w_2. This case is depicted in Figure 13a. On the other hand, if e_P is applied before P', let e_P send x_t from w_{t-1} to w_2 and let $P' := w_1 w_2 w_t$ as depicted in Figure 13b. Either case has the same effect on the underlying clusters as the original path P.

Therefore, we decompose each path P from D_P with more than three vertices in this manner, adding the resulting cycle P_Y to D_Y and replacing P with P' in D_P. All paths in D_P then have at most three vertices and Corollary 2.8 implies that we can completely integrate any of these paths with a disjoint cycle cover from D_Y in only two moves. Note also that each vertex in a cycle P_Y is an interior vertex of the original path P. Hence, even after introducing these additional cycles to D_Y, the shared degree of each vertex in D_Y remains at most the shared degree of that vertex in the original clustering-difference graph. This allows us to improve upon the distance bound given in Lemma 3.3.

The challenge in this approach lies in the fact that the interior vertex of P' (either w_{t-1} or w_2) and the label of e_P (either x_2 or x_t) depend on the order in which the corresponding transfers are applied. If a cycle cover $\mathcal{Y}$ does not include e_P, then integrating P' with $\mathcal{Y}$ is straightforward – one simply chooses the correct interior vertex for P' depending on

whether or not e_P has been applied yet. Once P' is applied, then if e_P remains in D_Y, its label is changed from x_t to x_2.

However, if e_P is contained in $\mathcal{Y}$, then we must make adjustments to the double-moves from Theorem 2.6 in order to take into account the different possible cases for e_P and P'. Nevertheless, this approach allows us to integrate sequential moves from D_P with disjoint cyclical moves from D_Y at no extra cost, resulting in the following greatly improved distance bound. The bound depends only on the larger of the second-largest shared degree and the overall change in cluster sizes rather than on the sum of these values as in Lemma 3.3.

Theorem 3.4. *Let $\mathcal{C}, \mathcal{C}'$ be k-clusterings of the same data set. Then*

$$d(\mathcal{C}, \mathcal{C}') \leq \eta_{i_1} + \max\left\{\eta_{i_2},\ \frac{1}{2}\sum_{i=1}^{k}\delta_i\right\},$$

where $\delta_i = |(|C_i| - |C_i'|)|$, η_i is the shared degree of c_i in $CDG(\mathcal{C}, \mathcal{C}')$, $i_1 = \arg\max \eta_i$, and $i_2 = \arg\max_{i \neq i_1} \eta_i$.

Proof. Let (D_P, D_Y) be any path-cycle decomposition of $D := CDG(\mathcal{C}, \mathcal{C}')$. For each path $P = w_1 ... w_t$ of the $\frac{1}{2}\sum_{i=1}^{k}\delta_i$ paths in D_P, if $t \geq 4$, we decompose P into a cycle P_Y and a short path P' as depicted in the cases of Figure 13. Specifically, let $P' := w_1 w_{t-1} w_t$ where the label of edge (w_1, w_{t-1}) is the item x_2 to be sent from w_1 to w_2 in P, as depicted by the blue edges in Figure 13a. Then P is replaced with P' in D_P. In addition, the cycle $P_Y = w_2 ... w_{t-1} w_2$ is introduced to D_Y, where the label of the artificial edge $e_P := (w_{t-1}, w_2)$ in P_Y is the item x_t to be sent from w_{t-1} to w_t in P, as depicted by the green edges in Figure 13b. Note that each vertex in P_Y is an interior vertex of the original path P; hence, in the resulting cycle graph D_Y, the shared degree of each vertex c_i remains at most η_i.

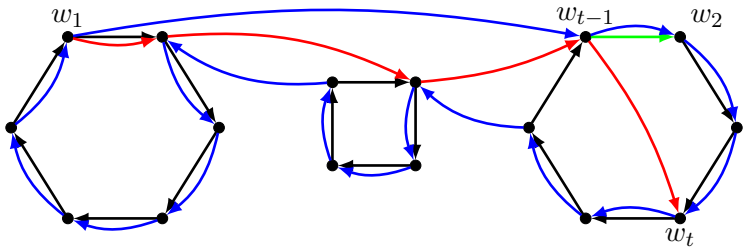
As in the proof of Lemma 3.3, we first apply at most $\eta_{i_1} - \eta_{i_2}$ cyclical moves to reduce the maximum shared degree in D_Y to at most η_{i_2}. Whenever an artificial edge e_P is applied in such a move, we change the interior vertex of the corresponding path P' in D_P so that $P' = w_1 w_2 w_t$ as in Figure 13b.

Now, again as in the proof of Lemma 3.3, we can reduce the maximum shared degree in D_Y by finding a disjoint cycle cover for the vertices of maximum shared degree and applying a double-move. However, in each such double-move, we will also integrate a path from D_P.

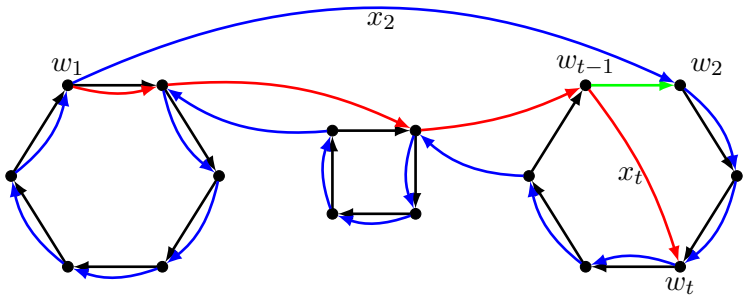
Let $\mathcal{Y}$ be such a set of disjoint cycles in D_Y, which can be found using the technique from [2]. We choose any path from D_P. Since each path in D_P has at most three vertices, if the selected path is an original path from $CDG(\mathcal{C}, \mathcal{C}')$, integrating the path with $\mathcal{Y}$ in a double-move is straightforward via Corollary 2.8. Again, if an artificial edge e_P is applied through this double-move and the corresponding path P' remains in D_P, we switch the interior vertex of P' from w_{t-1} to w_2 as in Figure 13b.

Hence, we assume the selected path from D_P is an introduced path of the form P' with corresponding artificial edge e_P. If e_P is not contained in the cycle cover $\mathcal{Y}$, integrating P' with $\mathcal{Y}$ is again straightforward via Corollary 2.8: the interior vertex of P' is known and we can simply apply one of the double-moves from Theorem 2.6. After the double-move is applied, we make any necessary adjustments to the remaining paths in D_P as in the previous paragraph. Additionally, if e_P remains in D_Y, we change its label from x_t to x_2 as in Figure 13a.

However, if the edge e_P corresponding to P' is contained in $\mathcal{Y}$, then we must make modifications to the double-moves of Theorem 2.6 to account for the two different cases for e_P and P'. Note that if this situation arises, e_P has not yet been applied so P' has the initial form $P' = w_1 w_{t-1} w_t$. We modify each of the four cases regarding the intersection points of P' with $\mathcal{Y}$ from Theorem 2.6 to perform the necessary transfers. Since e_P is included in $\mathcal{Y}$, both w_{t-1} and w_2 are necessarily covered by $\mathcal{Y}$, but w_1 and w_t need not be covered. Several of the case modifications depend on whether or not these two vertices are actually covered by the cycles.



(a) Original double-move from Case 1 of Theorem 2.6.



(b) A modified double-move for Case 1.

Figure 14: The original and modified double-move for Case 1.

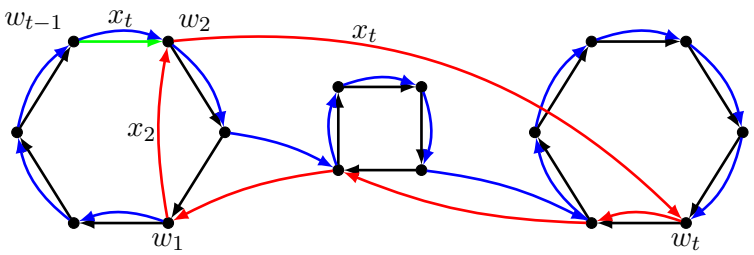
Case 1: All three vertices of $P' = w_1 w_{t-1} w_t$ are covered by $\mathcal{Y}$, where w_1 and w_t belong to different cycles of $\mathcal{Y}$ and w_{t-1} (and hence, also e_P and w_2) belongs to the same cycle as w_t. See the examples in Figure 14 – the artificial edge e_P is given by the green edge from w_{t-1} to w_2 and the other edges of $\mathcal{Y}$ are given in black. Note that we cannot simply apply the double-move from Case 1 in Theorem 2.6 as depicted for this scenario in Figure 14a (compare to Figure 9). In the first cyclical move, the edge e_P would be applied, sending the item x_t from w_{t-1} to w_2. Hence, w_{t-1} would then be unable to send x_t to w_t in the second sequential move.

We can address this by making a slight modification to this first cyclical move: instead of sending x_2 from w_1 to w_{t-1} and then following e_P, the move simply sends x_2 directly from w_1 to w_2. Then x_t remains at w_{t-1}, and in the second sequential move, x_t is correctly sent from w_{t-1} to w_t as seen in Figure 14b.

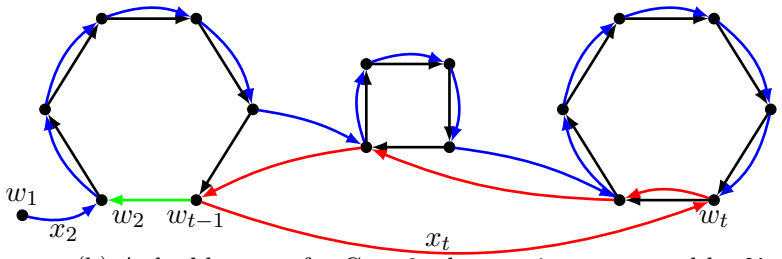
Note that in this modified double-move, the artificial edge e_P from w_{t-1} to w_2 is never actually applied. However, its intended purpose is accomplished: item x_2 is correctly received by w_2 from w_1, and x_t is correctly sent from w_{t-1} to w_t. Therefore, after the double-move is applied, we can remove P' from D_P and e_P along with the other edges of $\mathcal{Y}$ from D_Y, as desired.

Case 2: Of the vertices of $P' = w_1 w_{t-1} w_t$ which are covered by $\mathcal{Y}$, the first and last belong to different cycles, and the second-to-last vertex covered by $\mathcal{Y}$ belongs to a different cycle than the last. There are three double-moves based on the double-move from Case 2 of Theorem 2.6 which can be used depending on whether or not w_1, w_t, or both w_1 and w_t are covered by $\mathcal{Y}$. Depictions of these moves are given in Figure 15.

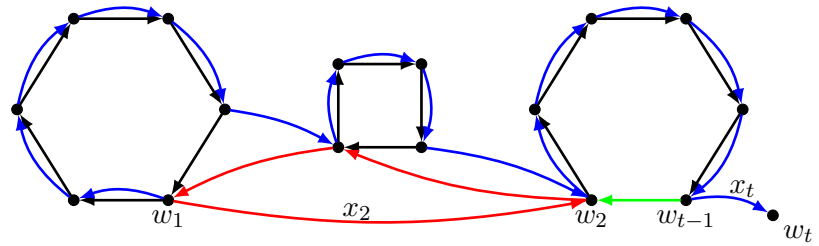
a) Vertices w_1 and w_t are both covered by $\mathcal{Y}$. In this situation, for Case 2 to apply, w_1 and w_t must belong to different cycles of $\mathcal{Y}$ and w_{t-1} must not belong to the same cycle as w_t. See Figure 15a. Then when performing the double-move from Case 2 of Theorem 2.6 as depicted in Figure 10, the edge e_P is applied in the first sequential move before any of the edges from P', sending x_t from w_{t-1} to w_2. Hence, if we switch the interior vertex of P' from w_{t-1} to w_2, we can apply this original



(a) A double-move for Case 2 when both w_1 and w_t are covered by $\mathcal{Y}$.



(b) A double-move for Case 2 when w_1 is not covered by $\mathcal{Y}$.



(c) A double-move for Case 2 when w_t is not covered by $\mathcal{Y}$.

Figure 15: Modified double-moves for Case 2.

double-move without any further modifications, as depicted in Figure 15a.

b) Vertex w_1 is not covered by $\mathcal{Y}$. Then for Case 2 to apply, w_{t-1} and w_t must belong to different cycles of $\mathcal{Y}$. See Figure 15b. As in Case 1, we cannot apply the original double-move since then w_{t-1} would be unable to send x_t to w_t in the second move. We can address this in the same way as in the modified double-move from Case 1: the first sequential move sends x_2 directly from w_1 to w_2, and then the second cyclical move sends x_t directly from w_{t-1} to w_t, as depicted in Figure 15b. Although e_P is never actually applied, all desired transfers are accomplished as in Case 1.

c) Vertex w_t is not covered by $\mathcal{Y}$. Then for Case 2 to apply, w_1 and w_{t-1} must belong to different cycles of $\mathcal{Y}$. See Figure 15c. We make a similar modification to that of the previous case: the first sequential move correctly sends x_t directly from w_{t-1} to w_t, and then the second cyclical move correctly sends x_2 directly from w_1 to w_2, as depicted in Figure 15c.

Case 3: Of the vertices of $P = w_1 w_{t-1} w_t$ which are covered by $\mathcal{Y}$, the first and last belong to the same cycle in $\mathcal{Y}$, while the second-to-last vertex belongs to a different cycle. The only scenario in which this case applies is when w_1 and w_t belong to the same cycle of $\mathcal{Y}$ and w_{t-1} belongs to a different cycle. We make a modification similar to the third double-move from the previous case. The first cyclical move sends x_t directly from w_{t-1} to w_t, and the second sequential move sends x_2 directly from w_1 to w_2. See Figure 16.

Case 4: All vertices of $P' = w_1 w_{t-1} w_t$ which are covered by $\mathcal{Y}$ belong to the same cycle. There are two double-moves, depicted in Figure 17, which can be used depending on whether or not w_1 is covered by $\mathcal{Y}$.

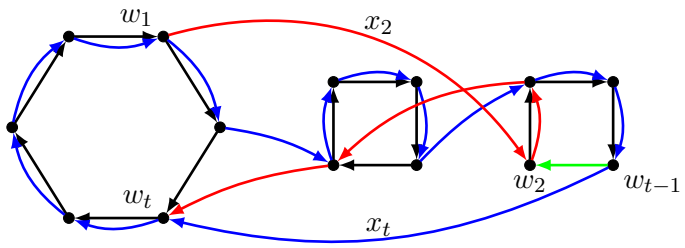
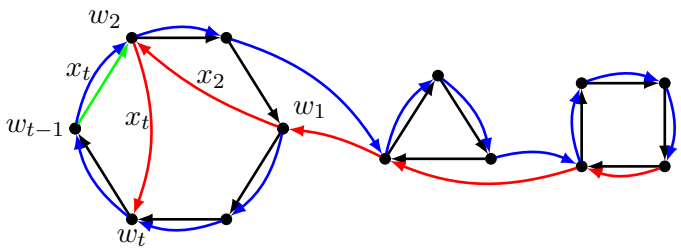
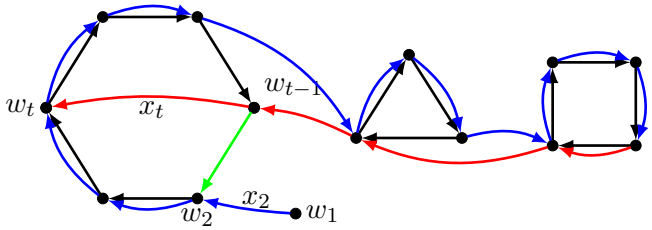


Figure 16: A modified double-move for Case 3.



(a) A double-move for Case 4 when w_1 is covered by $\mathcal{Y}$.



(b) A double-move for Case 4 when w_1 is not covered by $\mathcal{Y}$.

Figure 17: Modified double-moves for Case 4.

a) Vertex w_1 is covered by $\mathcal{Y}$. As in the first double-move for Case 2, in the original double-move for Case 4 of Theorem 2.6 the edge e_P is applied before any of the edges from P'. Hence, if we switch the interior vertex of P' from w_{t-1} to w_2, we can apply the double-move without any further modifications, as depicted in Figure 17a. Note that w_t may or may not be covered by the cycle containing w_1 and e_P.

b) Vertex w_1 is not covered by $\mathcal{Y}$. We make a modification similar to that of the second double-move for Case 2: the first sequential move directly sends x_2 from w_1 to w_2, and the second sequential move directly sends x_t from w_{t-1} to w_t. See Figure 17b. Note again that w_t may or may not be covered by the cycle containing e_P.

In each case, we are able to integrate P' with $\mathcal{Y}$ and apply all necessary transfers in only two moves. Therefore, at most η_{i_2} double-moves are needed to reduce the shared degree of D_Y to zero, and through each of these double-moves, we remove one of the $\frac{1}{2}\sum_{i=1}^{k}\delta_i$ paths from D_P. Afterwards, we may simply apply the remaining paths in D_P, if any, individually. The total number of moves used to transform $\mathcal{C}$ into $\mathcal{C}'$ is thus at most

$$(\eta_{i_1} - \eta_{i_2}) + 2\eta_{i_2} + \max\left\{\frac{1}{2}\sum_{i=1}^{k}\delta_i - \eta_{i_2},\ 0\right\} = \eta_{i_1} + \max\left\{\eta_{i_2},\ \frac{1}{2}\sum_{i=1}^{k}\delta_i\right\}.$$

□

4. Circuit Diameter of Partition Polytopes. A fundamental open question in linear programming is whether or not there exists a polynomial pivot rule for the simplex method. The existence of such a pivot rule would require that the *polynomial Hirsch conjecture* [20]

holds; i.e., that the combinatorial diameter of a polyhedron can be polynomially bounded. A recent effort to better understand the combinatorial diameter of polyhedra has been the study of the related circuit diameter [6, 8, 10, 19]. Whereas the original Hirsch conjecture is false in general [21, 26], the related *Circuit Diameter Conjecture* [6] remains open.

Recall that the circuits of the bounded-size partition polytope BPP correspond to cyclical and sequential moves of items among clusters. Therefore, as long as no cluster size constraints are violated during a clustering transformation, any resulting bounds on the transformation distance between clusterings have implications on the circuit distance between vertices in BPP. As a 0/1-polytope, the combinatorial diameter (and hence, also the circuit diameter) of BPP satisfies the Hirsch conjecture [24] – specifically, the combinatorial diameter is at most the number of items n. In this section, we will use the results from Section 3 to achieve much better upper bounds on the circuit diameter.

For the fixed-size partition polytope, Proposition 2.3 can be used to show that the combinatorial diameter is at most $\kappa_1 + \kappa_2$, where κ_1, κ_2 are the two largest fixed cluster sizes [2]. We begin by generalizing this bound to the circuit diameter of the bounded-size partition polytope by also taking into account the largest possible change in cluster sizes. Although we do not yet utilize any double-moves which integrate sequential and cyclical moves (see the upcoming Theorem 4.2), the bound of the following lemma is already better than the naive bound achieved by simply counting the sequential and cyclical moves separately – we can relate the shared degree of a vertex in a CDG to the change in size of the corresponding cluster.

Lemma 4.1. *For a bounded-size partition polytope $BPP(\kappa^+, \kappa^-)$, assume the corresponding clusters are indexed so that $\kappa_1^+ \geq \cdots \geq \kappa_k^+$ and let i_1, i_2 denote the two indices minimizing $\kappa_i^+ - \kappa_i^-$. Then the circuit diameter of $BPP(\kappa^+, \kappa^-)$ is at most*

$$\kappa_1^+ + \kappa_2^+ + \frac{1}{2} \sum_{i \neq i_1, i_2}^{k} (\kappa_i^+ - \kappa_i^-).$$

Proof. Let $\mathcal{C}, \mathcal{C}'$ be k-clusterings corresponding to vertices y, y' of $BPP(\kappa^+, \kappa^-)$. We can transform $\mathcal{C}$ into $\mathcal{C}'$ by separately applying sequential moves followed by cyclical double-moves in the manner of Lemma 3.3. All intermediate clusterings in this transformation satisfy the cluster size bounds of $BPP(\kappa^+, \kappa^-)$, so the process indeed corresponds to a circuit walk from y to y' in $BPP(\kappa^+, \kappa^-)$.

Let η_i denote the shared degree of vertex c_i in $CDG(C, C')$, and let $\delta_i := |(|C_i| - |C_i'|)|$. Lemma 3.3 then implies that the circuit distance from y to y' in $BPP(\kappa^+, \kappa^-)$ is at most

$$\eta_{j_1} + \eta_{j_2} + \frac{1}{2} \sum_{i=1}^{k} \delta_i, \tag{4.1}$$

where j_1, j_2 maximize η_i over all $i = 1, ..., k$. Trivially, for $i = 1, ...k$, it holds that $\eta_i \leq \kappa_i^+$ and $\delta_i \leq \kappa_i^+ - \kappa_i^-$. Hence, we obtain the following upper bound on the circuit diameter of $BPP(\kappa^+, \kappa^-)$ as a natural implication of Lemma 3.3:

$$\kappa_{j_1}^+ + \kappa_{j_2}^+ + \frac{1}{2} \sum_{i=1}^{k} (\kappa_i^+ - \kappa_i^-).$$

Note however that this bound can be immediately improved. For $i = 1, ..., k$, we must have $\eta_i + \delta_i \leq \kappa_i^+$ since $\eta_i + \delta_i$ is equal to the maximum of the indegree and outdegree of c_i. Rearranging this inequality yields $\eta_i + \frac{1}{2}\delta_i \leq \kappa_i^+ - \frac{1}{2}\delta_i$. Substituting into (4.1), we obtain

the following upper bound on the circuit distance from y to y':

$$\eta_{j_1} + \eta_{j_2} + \frac{1}{2}\sum_{i=1}^{k} \delta_i = \sum_{i=j_1,j_2} \left(\eta_i + \frac{1}{2}\delta_i\right) + \frac{1}{2}\sum_{i\neq j_1,j_2}^{k} \delta_i$$
$$\leq \sum_{i=j_1,j_2} \left(\kappa_i^+ - \frac{1}{2}\delta_i\right) + \frac{1}{2}\sum_{i\neq j_1,j_2}^{k} \delta_i.$$

Note that $\sum_{i=j_1,j_2}\left(\kappa_i^+ - \frac{1}{2}\delta_i\right) \leq \kappa_{j_1}^+ + \kappa_{j_2}^+ \leq \kappa_1^+ + \kappa_2^+$. Similarly, it holds that $\frac{1}{2}\sum_{i\neq j_1,j_2}^{k}\delta_i \leq \frac{1}{2}\sum_{i\neq j_1,j_2}^{k}(\kappa_i^+ - \kappa_i^-) \leq \frac{1}{2}\sum_{i\neq i_1,i_2}^{k}(\kappa_i^+ - \kappa_i^-)$. Thus, we obtain the stated bound. □

As in Theorem 3.4, we can significantly improve upon this diameter bound by using the double-moves from Theorem 2.6 to integrate sequential moves with sets of disjoint cyclical moves. Note that we must take care when applying these double-moves to bounded-size clusterings – certain moves require the existence of a vertex whose cluster size can be temporarily increased as demonstrated in Figure 7. Nevertheless, under a mild assumption on the slack in the cluster size constraints, we can ensure the existence of such a vertex through a simple pre-processing of the clusters. This assumption takes the form $\sum_{i=1}^{k}\kappa_i^+ > n + k - 2$, i.e., the sum of upper cluster size bounds exceeds the number n of items in the data set by at least $k-1$. Since in most clustering applications, the number of items is much larger than the number of clusters ($n \gg k$), we see this assumption as quite natural and not particularly restrictive. For example, it is satisfied as long as there is at least some slack in the size constraints for all but at most one cluster. A counterexample to this condition would, in particular, require that there are always at least two clusters whose sizes are at their upper bounds regardless of how the items are distributed.

Hence, we obtain the following improved diameter bound as an implication of the transformation distance bound from Theorem 3.4, which depends on the maximum of the second-largest cluster size and the largest possible change in cluster sizes.

Theorem 4.2. *For a bounded-size partition polytope $BPP(\kappa^+, \kappa^-)$, assume the corresponding clusters are indexed so that $\kappa_1^+ \geq \cdots \geq \kappa_k^+$ and let i_1 denote the index minimizing $\kappa_i^+ - \kappa_i^-$. If $\sum_{i=1}^{k}\kappa_i^+ > n + k - 2$ and if $\kappa_i^+ > \kappa_i^-$ for $i \neq i_1$, the circuit diameter of $BPP(\kappa^+, \kappa^-)$ is at most*

$$\kappa_1^+ + \max\left\{\kappa_2^+,\ \frac{1}{2}\sum_{i\neq i_1}^{k}(\kappa_i^+ - \kappa_i^-)\right\} + 2(k-2).$$

Proof. Let $\mathcal{C}, \mathcal{C}'$ be k-clusterings corresponding to vertices y, y' of $BPP(\kappa^+, \kappa^-)$. We can transform $\mathcal{C}$ into $\mathcal{C}'$ in the manner of Theorem 3.4. However, in order for all intermediate clusterings to satisfy the bounds of $BPP(\kappa^+, \kappa^-)$, we must make sure that when applying any version of the double-move from Case 4 of Theorem 2.6, there exists a suitable choice for u_s whose corresponding cluster size is strictly less than its upper bound and can be temporarily increased.

To ensure that this is always the case, we pre-process $\mathcal{C}$ and $\mathcal{C}'$ in the following manner. If there exists more than one cluster C_i (or C_i' in the case of $\mathcal{C}'$) such that $|C_i| = \kappa_i^+$, we choose such an index j with $|C_j| = \kappa_j^+ > \kappa_j^-$, which is possible since at most one index i satisfies $\kappa_i^+ = \kappa_i^-$. We then transfer any item from C_j to a different cluster C_ℓ which satisfies $|C_\ell| < \kappa_\ell^+ - 1$. Such an index ℓ must exist, else it would hold that

$$\sum_{i=1}^{k}|C_i| \geq 2 + \sum_{i=1}^{k}(\kappa_i^+ - 1) > 2 + (n + k - 2) - k = n.$$

This process is repeated at most $k-2$ times until the sizes of all clusters are less than their upper bounds – with the exception of at most one cluster.

After pre-processing both $\mathcal{C}$ and $\mathcal{C}'$ in this manner, let $\overline{\mathcal{C}}$ denote the clustering obtained after processing $\mathcal{C}$, let $\overline{\mathcal{C}'}$ denote the corresponding clustering for $\mathcal{C}'$, let j_1 denote the index whose cluster size is potentially at its upper bound in $\overline{\mathcal{C}}$, and let j_2 denote the corresponding index for $\overline{\mathcal{C}'}$. We now transform $\overline{\mathcal{C}}$ into $\overline{\mathcal{C}'}$ via the procedure in the proof of Theorem 3.4; however, if $j_1 \neq j_2$, we choose the first double-move that integrates a path P' from the path-cycle decomposition of $CDG(\overline{\mathcal{C}}, \overline{\mathcal{C}'})$ (in other words, the first double-move that alters any cluster sizes) such that the head of P' is c_{j_1}. This double-move then reduces the size of cluster j_1, and the only cluster whose size is potentially at its upper bound at any point throughout the remainder of the procedure is cluster j_2. This is due to the fact that although a cluster's size may change throughout the application of the double-moves, it remains between its size in $\overline{\mathcal{C}}$ and its size in $\overline{\mathcal{C}'}$.

Therefore, when transforming $\overline{\mathcal{C}}$ to $\overline{\mathcal{C}'}$, it will always possible to choose u_s in Case 4 of Theorem 2.6 such that the corresponding cluster size can be temporarily increased when performing the double-move. To see this, note that the cycle from which u_s is chosen covers at least two vertices, and at least one of these vertices must have a corresponding cluster size less than its upper bound. This holds since in the first double-move which alters cluster sizes as discussed above, c_{j_1} must not be contained in the cycle, and in the remaining double-moves, only the size of cluster j_2 could possibly be at its upper bound. Lastly, note that the choice of the vertex u_s is unaffected by the modifications in Case 4 of Theorem 3.4.

By Theorem 3.4, the total number of moves needed to transform $\overline{\mathcal{C}}$ into $\overline{\mathcal{C}'}$ is at most

$$\eta_{j_1} + \max\left\{\eta_{j_2},\ \frac{1}{2}\sum_{i=1}^{k}\delta_i\right\},$$

where j_1, j_2 denote the two indices maximizing the shared degree η_i in $CDG(\overline{\mathcal{C}}, \overline{\mathcal{C}'})$. As in the proof of Lemma 4.1, since $\eta_{j_1} + \frac{1}{2}\delta_{j_1} \leq \kappa^+_{j_1} - \frac{1}{2}\delta_{j_1}$, this bound is at most

$$\begin{aligned}
\max\left\{\eta_{j_1} + \eta_{j_2},\ \left(\eta_{j_1} + \frac{1}{2}\delta_{j_1}\right) + \frac{1}{2}\sum_{i\neq j_1}^{k}\delta_i\right\} &\leq \max\left\{\eta_{j_1} + \eta_{j_2},\ \left(\kappa^+_{j_1} - \frac{1}{2}\delta_{j_1}\right) + \frac{1}{2}\sum_{i\neq j_1}^{k}\delta_i\right\} \\
&\leq \max\left\{\kappa^+_{j_1} + \kappa^+_{j_2},\ \kappa^+_{j_1} + \frac{1}{2}\sum_{i\neq j_1}^{k}(\kappa^+_i - \kappa^-_i)\right\} \\
&\leq \kappa^+_1 + \max\left\{\kappa^+_2,\ \frac{1}{2}\sum_{i\neq i_1}^{k}(\kappa^+_i - \kappa^-_i)\right\}.
\end{aligned}$$

Since all intermediate clusterings in this transformation satisfy the cluster size constraints of $BPP(\kappa^+, \kappa^-)$, the transformation indeed corresponds to a circuit walk in $BPP(\kappa^+, \kappa^-)$. Taking into account the at most $2(k-2)$ additional circuit steps needed for the pre-processing of $\mathcal{C}$ and $\mathcal{C}'$, we obtain the stated improved bound. □

5. Conclusions and Future Directions. In this work, we provide methods based on linear programming and network theory for transforming k-clusterings using sequences of cyclical and sequential moves of items among clusters. This leads to upper bounds on the transformation distance between two general k-clusterings as well as the circuit diameter of the bounded-size partition polytope. There are several natural directions for future research in this area.

First, we note that although we can bound the transformation distance between clusterings based on properties of their CDG, computing the exact transformation distance appears to be quite challenging. A first step toward better understanding this open problem would be to determine its computational complexity.

Next, we prove in Theorem 4.2 an upper bound on the circuit diameter of the bounded-size partition polytope using the transformation distance bound from Theorem 3.4 and

modified double-moves from Theorem 2.6 which integrate sequential moves of items with cyclical moves. A subsequent research question is whether or not we can also bound the combinatorial diameter of the polytope in such a manner. The edges of BPP have a more technical characterization than its circuits – only certain cyclical and sequential moves actually correspond to edges between vertices [12]. However, through a careful ordering of cyclical and sequential moves and double-moves, we believe new bounds on the combinatorial distance between vertices in the polytope could be achievable.

Additionally, in Theorem 3.4, we use an *arbitrary* path-cycle decomposition (D_P, D_Y) of the clustering-difference graph $D := CDG(\mathcal{C}, \mathcal{C}')$ to bound the transformation distance between the clusterings. It is possible to instead construct a decomposition exhibiting potentially useful properties. For instance, solving a minimum-cost circulation problem over D yields a decomposition in which D_Y has a maximum number of edges. Modifying this circulation problem can yield a decomposition in which the maximum shared degree in D_P is minimized. Through further analysis, these extremal choices for the path-cycle decomposition might lead to better upper bounds on the transformation distance.

Finally, we note that the transformation distance $d(\mathcal{C}, \mathcal{C}')$ is formally a metric. Hence, if we are able to compute $d(\mathcal{C}, \mathcal{C}')$, we can interpret it as a measure of the distance between given k-clusterings of the same data set. There is significant interest in comparing clusterings in the literature [15, 23]. However, most measures typically do not take into account the potential labels of the clusters and are instead based on pairwise relationships among the items. Our new metric takes a fundamentally different approach to measuring the difference between clusterings, motivating a comparative study.

Acknowledgements. Borgwardt gratefully acknowledges support of this work through NSF award 2006183 *Circuit Walks in Optimization*, Algorithmic Foundations, CCF, Division of Computing and Communication Foundations, and through Simons Collaboration Grant 524210 *Polyhedral Theory in Data Analytics* before.

REFERENCES

[1] M. Balinski and A. Russakoff, *On the assignment polytope*, SIAM Review, 16 (1974), pp. 516–525.
[2] S. Borgwardt, *On the diameter of partition polytopes and vertex-disjoint cycle cover*, Mathematical Programming, Ser. A, 141 (2013), pp. 1–20.
[3] S. Borgwardt, A. Brieden, and P. Gritzmann, *Mathematics in agriculture and forestry: Geometric clustering for land consolidation*, IFORMS news, Dec. issue (2013).
[4] S. Borgwardt, A. Brieden, and P. Gritzmann, *Geometric clustering for the consolidation of farmland and woodland*, The Mathematical Intelligencer, 36 (2014), pp. 37–44.
[5] S. Borgwardt, A. Brieden, and P. Gritzmann, *Geometrisches Clustering: Mathematik für die Flurverbesserung (Geometric clustering: Mathematics for land improvement)*, Mitteilungen der DMV, 23 (2015), pp. 82–90.
[6] S. Borgwardt, E. Finhold, and R. Hemmecke, *On the circuit diameter of dual transportation polyhedra*, SIAM Journal on Discrete Mathematics, 29 (2016), pp. 113–121.
[7] S. Borgwardt and F. Happach, *Good Clusterings Have Large Volume*, Operations Research, 67 (2019), pp. 215–231.
[8] S. Borgwardt, J. A. D. Loera, and E. Finhold, *Edges vs circuits: a hierarchy of diameters in polyhedra*, Advances in Geometry, 16 (2016), pp. 511–530.
[9] S. Borgwardt and S. Onn, *Efficient solutions for weight-balanced partitioning problems*, Discrete Optimization, 21(C) (2016), pp. 71–84.
[10] S. Borgwardt, T. Stephen, and T. Yusun, *On the circuit diameter conjecture*, Discrete & Computational Geometry, 60 (2018), pp. 558–587.
[11] S. Borgwardt and C. Viss, *A polyhedral model for enumeration and optimization over the set of circuits*, Discrete Applied Mathematics, (in press, 2019).
[12] S. Borgwardt and C. Viss, *Circuit Walks in Integral Polyhedra*, Discrete Optimization, (in press, 2020).
[13] J. A. De Loera, R. Hemmecke, and J. Lee, *On augmentation algorithms for linear and integer-linear programming: from Edmonds-Karp to Bland and beyond*, SIAM Journal on Optimization, 25 (2015), pp. 2494–2511.
[14] J. A. De Loera, E. D. Kim, S. Onn, and F. Santos, *Graphs of transportation polytopes*, Journal

of Combinatorial Theory, Ser. A, 116 (2009), pp. 1306–1325.
[15] A. J. Gates, I. B. Wood, W. P. Hetrick, and Y. Ahn, *Element-centric clustering comparison unifies overlaps and hierarchy*, Scientific Reports, 9 (2019), p. 1.
[16] J. E. Graver, *On the foundation of linear and integer programming I*, Mathematical Programming, 9 (1975), pp. 207–226.
[17] R. Hemmecke, S. Onn, and L. Romanchuk, *N-fold integer programming in cubic time*, Mathematical Programming, Ser. A, 137 (2013), pp. 325–341.
[18] R. Hemmecke, S. Onn, and R. Weismantel, *A polynomial oracle-time algorithm for convex integer minimization*, Mathematical Programming, Ser. A, 126 (2011), pp. 97–117.
[19] S. Kafer, K. Pashkovich, and L. Sanità, *On the circuit diameter of some combinatorial polytopes*, SIAM Journal on Discrete Mathematics, 33 (2017).
[20] E. D. Kim and F. Santos, *An update on the Hirsch conjecture*, Jahresbericht der Deutschen Mathematiker-Vereinigung, 112 (2010), pp. 73–98.
[21] V. Klee and D. W. Walkup, *The d-step conjecture for polyhedra of dimension $d < 6$*, Acta Mathematica, 133 (1967), pp. 53–78.
[22] V. Klee and C. Witzgall, *Facets and vertices of transportation polyhedra*, in Mathematics of the decision sciences, vol. 1, American Mathematical Society, 1968, pp. 257–282.
[23] M. Meilă, *Comparing clusterings—an information based distance*, Journal of Multivariate Analysis, 98 (2007), pp. 873 – 895.
[24] D. Naddef, *The Hirsch Conjecture is true for $(0, 1)$-polytopes*, Mathematical Programming, 45 (1989), pp. 109–110.
[25] R. T. Rockafellar, *The elementary vectors of a subspace of $\mathbb{R}^N$*, in Combinatorial Mathematics and its Applications, 1969, pp. 104–127.
[26] F. Santos, *A counterexample to the Hirsch conjecture.*, Annals of Mathematics, 176 (2011), pp. 383–412.