Identifying Imperfect Clones in Elections

Piotr Faliszewski[1], Łukasz Janeczko[1], Grzegorz Lisowski[2], Kristýna Pekárková[1], Ildikó Schlotter[3,4]

[1]AGH University of Kraków, Poland
[2]University of Groningen, The Netherlands
[3]ELTE Centre for Economic and Regional Studies, Hungary
[4]Budapest University of Technology and Economics, Hungary
faliszew@agh.edu.pl, ljaneczk@agh.edu.pl, g.a.lisowski@rug.nl, pekarkova@agh.edu.pl, schlotter.ildiko@krtk.elte.hu

Abstract

A perfect clone in an ordinal election (i.e., an election where the voters rank the candidates in a strict linear order) is a set of candidates that each voter ranks consecutively. We consider different relaxations of this notion: *independent* or *subelection clones* are sets of candidates that only some of the voters recognize as a perfect clone, whereas *approximate clones* are sets of candidates such that every voter ranks their members close to each other, but not necessarily consecutively. We establish the complexity of identifying such imperfect clones, and of partitioning the candidates into families of imperfect clones. We also study the parameterized complexity of these problems with respect to a set of natural parameters such as the number of voters, the size or the number of imperfect clones we are searching for, or their level of imperfection.

1 Introduction

Let us consider an ordinal election, with a given set of candidates and a collection of voters that rank these candidates. Such elections may appear in classic political settings where the goal is to elect a president or some other leader, but also in various other contexts, such as sport events where the candidates are athletes and "voters" are particular competitions, ranking them according to their performance. Ordinal elections also arise from multicriteria evaluations where the "voters" are the applied quality measures, e.g., when listing the best universities or the most livable cities; see the work of Boehmer and Schaar (2023) for such election data. In each of these settings, we expect some groups of candidates to be similar to each other. For example, left- or right-wing politicians are likely to have similar platforms, athletes with similar abilities—such as, e.g., the top sprinters in Tour de France—typically perform similarly, and universities of a particular type—such as small liberal arts colleges—have similar features. Consequently, we expect voters to rank such candidates close to each other. Formally, sets of candidates that all voters rank on consecutive positions are called *clones* and were already studied in some detail both in AI and (computational) social choice; as examples, we point to the works of Tideman (1987), Laslier (1996, 2000) and Elkind, Faliszewski, and Slinko (2011, 2012); see also the notion of *composition consistency* (Laffond, Laine, and Laslier 1996; Laslier 1997) and its connection to clones (Berker et al. 2025). Yet, *similar* does not mean *identical* and, so, expecting all the voters to perfectly recognize all the clones—i.e., rank them consecutively—seems overly demanding. Hence, we focus on the problem of identifying those candidates that form *imperfect clones*, and on the problem of partitioning the set of candidates into such clones. We believe that the ability to solve these problems allows one to better understand the structure and nature of the candidates in a given election; in this sense, we follow up on the work of Janeczko et al. (2024) on identifying hidden structures in voters' preferences. Our work is also related to that of Procaccia, Schiffer, and Zhang (2024), who consider approximate clones in the context of training LLMs. However, in contrast to their work, our focus is on establishing the (parameterized) complexity of our problems.

There are two main ways in which clones can be imperfect. First, we may require all the voters to recognize all the clones, but cut them some slack on ranking their members consecutively: Voters simply need to rank clone members on close enough positions, but they are allowed to put some candidates that are not members of the given clone in between. Second, we may require the voters that recognize the clones to indeed rank their members consecutively, but allow each clone to be recognized by a possibly different subset of the voters. We refer to the former type of clones as *approximate* and to the latter one as *independent*. If we insist that all the independent clones are recognized by the same voters, then we refer to them as *subelection* clones. We find that the complexity of our clone identification/clone partition problems very strongly depends on the type of imperfection that we consider. For example, we obtain the following results:

1. There is a simple, polynomial-time algorithm for identifying independent clones, but recognizing approximate clones is NP-hard. Yet, if members of approximate clones are not ranked too far apart, then we can identify them using an FPT algorithm (see Section 3).
2. Partitioning the set of candidates into a given number of imperfect clones, each of at most a given size, is NP-hard, but there are algorithms for some special cases. For example, there is an FPT algorithm for approximate clones for parameterization by their imperfection level, an XP algorithm for independent/subelection clones for

parameterization by the number of clones, and an FPT algorithm for subelection clones for parameterization by the number of voters (see Section 4 and Table 1).

Indeed, we classify the parameterized complexity of the partitioning problem for all our basic parameters that clone partitions may have. The proofs of results marked by a $\star$ symbol are deferred to the full version (Faliszewski et al. 2025).

2 Preliminaries

For an integer k, we write $[k]$ for the set $\{1, \ldots, k\}$.

Elections. An *(ordinal) election* is a pair $E = (C, V)$, where $C = \{c_1, \ldots, c_m\}$ is a set of *candidates* and $V = \{v_1, \ldots, v_n\}$ is a set of *voters* (sometimes also referred to as *votes*). Each voter v has a *preference order* $\succ_v$ that strictly ranks all candidates in C. We write $v\colon a \succ b$ to indicate that v ranks candidate a above candidate b, and we use analogous notation for larger candidate sets. A set of candidates in a preference order should be interpreted as ranking its members in some arbitrary but fixed order. For a voter v and candidate c, we write $\mathrm{pos}_v(c)$ to denote the position of c in v's preference order; the top-ranked candidate has position 1.

Clone Structures and Clone Partitions. Let $E = (C, V)$ be an election and let $b \geq 1$ be an integer. We say that a nonempty subset X of candidates is a *b-perfect clone* in E if $|X| \leq b$ and each voter ranks members of X consecutively (but, possibly, in a different order). We are also interested in clones that are in some way imperfect, i.e., either not all voters recognize the given candidates as forming a clone (so *independent* groups of voters may recognize different clones), or the voters do not agree on the exact compositions of the clones (so the clones are *approximate*). We formalize these ideas below:

Approximate Clones. Let $b \geq 1$ and $q \geq 0$ be integers. A subset X of candidates is a *(q, b)-approximate clone* if $|X| \leq b$ and for each voter $v \in V$ we have $\max_{x \in X} \mathrm{pos}_v(x) - \min_{x \in X} \mathrm{pos}_v(x) \leq q + b - 1$. In other words, each voter ranks the candidates from X *almost* consecutively, ranking at most q candidates outside X in between any two candidates from X.

Independent Clones. Let b and w be positive integers. A subset X of candidates is a *(w, b)-independent clone* if $|X| \leq b$ and at least w voters *recognize* X, i.e., rank all members of X consecutively. Janeczko et al. (2024) studied such sets under the name *hidden clones*.

A q-approximate clone or a w-independent clone is a clone that is (q, b)-approximate or (w, b)-independent for some b, respectively. A *{perfect, q-approximate, w-independent}-clone structure* for election E is the set of all {perfect, q-approximate, w-independent}-clones that appear in E. A *{perfect, q-approximate, w-independent}-clone partition* for E is a family $\mathcal{C} = \{C_1, \ldots, C_t\}$ of subsets of C whose members are {perfect, q-approximate, w-independent}-clones and each candidate belongs to exactly one C_i. We also consider subelection-clone partitions:

Subelection Clones. Let b and w be positive integers. Then $(C_1, \ldots, C_t)$ is a *subelection-clone partition* if there is a subset $W \subseteq V$ of at least w voters such that $(C_1, \ldots, C_t)$ is a perfect-clone partition for election (C, W) with each clone C_i having size at most b. We refer to the members of such a partition as *(w, b)-subelection clones*.

We often speak of clones, clone structures, and clone partitions without specifying the exact type of clones involved. In such cases, the relevant type is clear from the context. Similarly, we sometimes speak of, e.g., (q, b)-approximate-clone partition to mean an approximate-clone partition whose each member is a (q, b)-approximate clone.

Remark 2.1. *The difference between independent clones and subelection clones is that two sets are (w, b)-independent clones if for each of them there is a group of w voters who ranks their members consecutively, but these two groups do not need to coincide and for small enough w may even be disjoint. On the other hand, subelection clones have to be ranked consecutively by the same w voters.*

Our theoretical results mostly regard the computational complexity of the following family of problems.

{PERFECT, APPROXIMATE, INDEPENDENT, SUBELECTION}-CLONE PARTITION

Input: An election $E = (C, V)$, positive integers b and t, as well as integer $q \geq 0$ for the APPROXIMATE variant, and integer $w \geq 1$ for SUBELECTION and INDEPENDENT variants.

Question: Is there a clone partition of size at most t for election E that consists of b-perfect, (q, b)-approximate, (w, b)-independent, or (w, b)-subelection clones, respectively?

Computational Complexity. We assume familiarity with classic and parameterized computational complexity theory, as presented in the textbooks of Papadimitriou (1994), Niedermeier (2006) and Cygan et al. (2015). The following problem plays an important role in our hardness proofs.

Definition 2.2. *The input of the* RESTRICTED EXACT COVER BY 3-SETS (RX3C) *problem, consists of a ground set X and a family $\mathcal{S}$ of size-3 subsets of X with each $x \in X$ occurring in exactly three sets of $\mathcal{S}$. We ask if there is a collection of $k = |X|/3$ sets from $\mathcal{S}$ whose union is X.*

Naturally, the k sets about whose existence we ask must be disjoint. This variant of the problem was shown to be NP-complete by Gonzalez (1985). Some of our proofs will rely on the 7-colorability of a certain graph underlying an instance of RXC3, as stated in the following observation.

Proposition 2.3. *Given an instance $(X, \mathcal{S})$ of RX3C, we can partition $\mathcal{S}$ in polynomial time into $\mathcal{S}_1, \ldots, \mathcal{S}_7$ so that for each $i \in [7]$ the sets in $\mathcal{S}_i$ are pairwise disjoint.*

Proof. Consider the graph H whose vertex set is $\mathcal{S}$ and where two triples from $\mathcal{S}$ are connected by an edge if and only if they share a common element of X. The maximum degree in H is at most 6, because for any $S \in \mathcal{S}$, there are at most two triples in $\mathcal{S} \setminus \{S\}$ containing a fixed element $x \in S$. Thus, we can find a proper coloring of H with 7 colors in polynomial time (using a greedy coloring). The 7 color classes obtained give the desired partitioning of $\mathcal{S}$. □

We will also use the following special variant of the EXACT COVER problem in some of our algorithms.

Definition 2.4. *In the* WEIGHTED EXACT COVER (WEC) *problem we are given a set $X = [m]$ of elements, a family $\mathcal{S} = \{S_1, \ldots, S_n\}$ of subsets of X where each set S_i is associated with weight w_i, and a nonnegative integer t. We ask if there is a set $I \subseteq [n]$ such that*

1. *for each $a, b \in I$, $a \neq b$, $S_a \cap S_b = \emptyset$,*
2. *$\bigcup_{\ell \in I} S_\ell = X$, and*
3. *$\sum_{\ell \in I} w_\ell \leq t$.*

Consider some positive integer m. A set $A \subseteq [m]$ is an *interval* if either $A = \emptyset$ or $A = \{i, i+1, i+2, \ldots, j\}$ for some integers i and j, $i \leq j$. Further, given a nonnegative integer q, we say that A is a *q-approximate interval* if the set

$$\{\min(A)+1, \min(A)+2, \ldots, \min(A)+(q-1)\} \cup A \cup \{\max(A)-1, \max(A)-2, \ldots, \max(A)-(q-1)\}$$

is an interval. Roughly speaking, a q-approximate interval is an interval that, possibly, is missing some of the first and last q values. The next result follows due to a simple dynamic programming, provided for the sake of completeness.

Proposition 2.5. *There is an algorithm that given an instance I of* WEC *where each set is a q-approximate interval decides in time $\mathcal{O}^*(2^{4q})$ if I is a yes-instance.*[1]

Proof. Consider an instance of WEC with ground set $[m]$, a family $\mathcal{S} = \{S_1, \ldots, S_n\}$ of sets where each S_i is a q-approximate interval of weight w_i, and an integer t. We define a function f such that, for each integer $i \in [m] \cup \{0\}$ and set $B \subseteq \{i+1, i+2 \ldots, i+2q\}$, $f(i, B)$ is the smallest number t' for which there is a set $I' \subseteq [n]$ such that:

1. for each $a, b \in I'$, $a \neq b$, $S_a \cap S_b = \emptyset$,
2. $\bigcup_{\ell \in I'} S_\ell = [i] \cup B$, and
3. $\sum_{\ell \in I'} w_\ell = t'$.

If these conditions cannot be met, then we let $f(i, B) = \infty$. As a base case, we observe that $f(0, \emptyset) = 0$. Consider integers $i, j \in [m]$, $i \leq j$, sets $B' \subseteq \{i+1, \ldots, i+2q\}$ and $B'' = \{j+1, \ldots, j+2q\}$, and a set S_ℓ. We say that (i, j, B', B'', S_ℓ) is *consistent* if:

1. $S_\ell \cap ([i] \cup B') = \emptyset$, and
2. $S_\ell \cup ([i] \cup B') = [j] \cup B''$.

In other words, if (i, j, B', B'', S_ℓ) is consistent and we have a family of sets whose union is $[i] \cup B'$, then we can extend this family with set S_ℓ so the union of the extended family is $[j] \cup B''$. Now we express $f(i, B)$ recursively as follows (we consider $j \in [m]$ and $B'' \subseteq \{j+1, \ldots, j+2q\}$):

$$f(j, B'') = \min_{\substack{(i, j, B', B'', S_\ell) \\ \text{is consistent}}} f(i, B') + w_\ell,$$

and we let $f(j, B'') = \infty$ if there is no consistent (i, j, B', B'', S_ℓ) to consider. Using this recursive formulation and standard dynamic programming techniques, we can compute $f(m, \emptyset)$ in time $\mathcal{O}^*(2^{4q})$; this running time stems from the fact that there are $\mathcal{O}^*(2^{2q})$ arguments that f can take, and our recursive formula requires us to consider $\mathcal{O}^*(2^{2q})$ consistent tuples. We accept if $f(m, \emptyset) \leq t$. The algorithm is correct because $\mathcal{S}$ consists of q-approximate intervals. □

[1] The notation $\mathcal{O}^*$ hides polynomial factors.

3 Complexity of Finding Clone Structures

For the case of perfect clones, Elkind, Faliszewski, and Slinko (2012) characterized what set families can appear as perfect-clone structures and provided simple algorithms for computing such structures. Similar algorithms also work for independent clones, as observed by Janeczko et al. (2024).

Proposition 3.1 (Janeczko et al. (2024))**.** *There are polynomial-time algorithms that, given an election E and an integer w, (a) test if a given set of candidates forms an independent clone recognized by at least w voters, and (b) compute a clone structure consisting of such independent clones.*

In contrast, deciding if there is a single q-approximate clone of a certain size is NP-complete.

Theorem 3.2. *Deciding if there exists a q-approximate clone of size exactly b in a given election is* NP*-complete.*

Proof. Containment in NP is clear, since we can verify in polynomial time whether a given set of candidates is a q-clone in the election. We present a reduction from the INDEPENDENT SET problem that, given an undirected graph $G = (U, F)$ and an integer k, asks whether G contains an independent set of size k. We let $U = \{u_1, \ldots, u_r\}$.

We introduce two dummies, d_1 and d_2, and for each vertex $u_i \in U$ we create a *vertex candidate* with the same name. We define $V = \{v_i \mid i \in [r]\} \cup \{w_f \mid f \in F\}$ as the set of voters, with preference orders as below (recall the convention about listing sets in preference orders):

$$v_i : d_1 \succ d_2 \succ U \setminus \{u_i\} \succ u_i,$$
$$w_f : d_2 \succ u_i \succ d_1 \succ U \setminus \{u_i, u_j\} \succ u_j$$

where $f = \{u_i, u_j\} \in F$ is an edge (we take $i < j$). We finish our construction by setting $b = k$ and $q = r - k$.

To see the correctness of the reduction, assume first that Q is a q-clone of size k in our election E. Notice that for each vertex candidate u_i, both d_1 and d_2 are at distance at least r from u_i in some preference order (namely, that of v_i). Consequently, and since $k > 2$, we get that $Q \subseteq U$. We claim that the k vertices in Q form an independent set in G. Indeed, assuming $u_i, u_j \in Q$ are connected by an edge of G, we know that u_i and u_j are at a distance $r+1$ in the preference order of voter w_f, for $f = \{u_i, u_j\} \in F$; a contradiction. Conversely, it is straightforward to check that an independent set of size k in G forms a q-clone in E. □

Fortunately, there are ways to circumvent this hardness result. Below we show an algorithm that decides if a given election has a size-b q-approximate clone in $\mathsf{FPT}(q)$ time.

Theorem 3.3. *There is an algorithm that given an election E and integers b and q decides in $\mathsf{FPT}(q)$ time if E contains a size-b q-approximate clone and, if so, returns one.*

Proof. Let election $E = (C, V)$ and integers b and q be our input. We give an algorithm that either finds a size-b q-approximate clone Q within E or concludes that no such clone exists. To do so, we employ the bounded search tree technique. The algorithm works as follows.

First, we fix an arbitrary voter $v \in V$ and guess candidates $c', c'' \in C$ such that, supposedly, v ranks c' highest and c'' lowest among the members of Q. Since Q is to be a size-b q-approximate clone, there must be some $q' \leq q$ such that:

$$\mathrm{pos}_v(c'') - \mathrm{pos}_v(c') + 1 = b + q',$$

that is, v must rank exactly $b + q'$ candidates between c' and c'' (inclusively). If this is not the case, then our guess was incorrect and we terminate the current computation path in the search tree without producing a result. We refer to each candidate $d \in C \setminus Q$ such that $v\colon c' \succ d \succ c''$ as an *intruder*. We must have exactly q' intruders. We form a set

$$Q_0 = \{c \in C \mid c' \succ_v c \succ_v c''\} \cup \{c', c''\}.$$

Naturally, $Q \subseteq Q_0$, and $Q_0 \setminus Q$ is exactly the set of intruders.

We next build a sequence $Q_0 \supseteq Q_1 \supseteq Q_2 \supseteq \cdots$ of sets where each Q_i has one intruder fewer than Q_{i-1}, and all constructed sets contain Q. Given Q_{i-1}, we compute Q_i as follows: If $|Q_{i-1}| < b$ then we terminate this computation path as no subset of Q_{i-1} can be a size-b q-approximate-clone. Otherwise, we check if there is a voter $u \in V$ for which u's most- and least-preferred candidates in Q_{i-1}, denoted a' and a'', satisfy

$$\mathrm{pos}_u(a'') - \mathrm{pos}_u(a') + 1 > b + q.$$

If no such voter u exists, then Q_{i-1} is a q-approximate clone, and all its size-b subsets are q-approximate clones as well. Thus, we output one of them arbitrarily, and terminate with answer *yes*. Otherwise, it is clear that at least one of a' and a'' is an intruder that we need to remove. We guess the intruder $\hat{a} \in \{a', a''\} \setminus Q$, and set $Q_i = Q_{i-1} \setminus \{\hat{a}\}$. This finishes the description of the algorithm.

The algorithm terminates at latest after forming the set $Q_{q'+1}$ which must have fewer than b candidates. The number of guesses that the algorithm makes is therefore at most

$$\underbrace{|C| \cdot (q+1)}_{\text{guesses of } c' \text{ and } c''} \cdot \underbrace{2^q}_{\substack{\text{guesses of} \\ \text{the intruders}}}.$$

Since each set Q_i can be obtained from Q_{i-1} in $\mathcal{O}(|C| \cdot |V|)$ time, we conclude that the algorithm runs in $\mathsf{FPT}(q)$ time. The correctness follows from the remarks made in the description of the algorithm. □

In fact, the algorithm that we use in the above proof can produce a succinct description of all size-b q-approximate clones that are subsets of a given set S of candidates such that $|S| \leq q + b$. To do so, whenever the algorithm is about to terminate with a *yes* answer, it outputs the then-considered set Q_{i-1} indicating that each of its size-b subsets is a q-approximate clone, and proceeds to the next path within the search tree instead of terminating.

Corollary 3.4. *There is an algorithm that given an election $E = (C, V)$, integers b and q, and a subset S of at most $b+q$ candidates from C outputs in $\mathsf{FPT}(q)$ time a family of sets $S_1, S_2, \ldots$ such that a set $Q \subseteq S$ is a size-b q-approximate clone if and only if it is a subset of some S_i.*

	Approximate clones	Independent clones	Subelection clones
$b = 2$	P (P4.2)	P (P4.2)	NP-c (T4.9)
$b = 3$	NP-c (T4.3)	NP-c (T4.6)	NP-c (T4.9)
param. t	paraNP-h (T4.3)	XP(t) (P4.7) W[1]-h (T4.8)	XP(t) (P4.10) W[1]-h (T4.12)
param. q	FPT(q) (T4.5)	–	–
$w = 2$	–	NP-c (T4.6)	P (P4.10)
param. w	–	NP-c (T4.6)	XP(w) (P4.10) W[1]-h (T4.12)
param. n	paraNP-h (T4.4)	paraNP-h (T4.6)	FPT(n)

Table 1: Computational complexity of finding partitions into imperfect clones. The cells marked with "–" refer to parameterizations undefined for the given problem. NP-c, W[1]-h, and paraNP-h stand for NP-completeness, W[1]-hardness, and para-NP-hardness, respectively.

4 Complexity of Finding Clone Partitions

In this section we discuss the complexity of our CLONE PARTITION family of problems for various types of clones. While for perfect clones the problem is easily seen to be solvable in polynomial time (indeed, this is essentially a folk result), for imperfect (i.e., approximate, independent, and subelection) clones the problem is NP-complete. We explore to what extent these intractability results can be circumvented using parameterized algorithms.

Proposition 4.1. PERFECT-CLONE PARTITION *is in* P.

Proof. Let our input consist of election $E = (C, V)$ over m candidates, together with integers b and t. We fix an arbitrary voter v and we rename the candidates so that

$$v\colon c_1 \succ c_2 \succ \cdots \succ c_m.$$

For each pair of integers $i, j \in [m]$, $i \leq j$, we test if $\{c_i, c_{i+1}, \ldots, c_j\}$ is a clone. We form an instance I of WEC with ground set $[m]$ and set family $\mathcal{S}$ where for each perfect clone $\{c_i, c_{i+1}, \ldots, c_j\}$ the set $\{i, i+1, \ldots, j\}$ is put into $\mathcal{S}$ with weight 1. We let t be the upper bound on the total weight of a solution for I. We solve I using Proposition 2.5 for $q = 0$, and accept if and only if I is a yes-instance. □

Before examining our three problems of partitioning the candidate set into imperfect clones in Sections 4.1, 4.2, and 4.3, we show that finding a partition into approximate or independent clones can be easily solved if the maximum allowed size of the clones is $b = 2$.

Proposition 4.2. APPROXIMATE- *and* INDEPENDENT-CLONES PARTITION *are in* P *for* $b = 2$.

Proof. Let our input be $E = (C, V)$ with integers b, q or w (depending on the problem), and t. We create a graph G over C where candidates c and c' are connected by an edge if they form a q-approximate clone or if they are seen as perfect clones by at least w voters, depending on the problem; note

that G can be constructed in polynomial time. Now, we find a maximum-size matching M in G. Clearly, our instance is a yes-instance exactly if $|M| + (|C| - 2|M|) \leq t$. □

4.1 Approximate Clones

Focusing now on APPROXIMATE-CLONES PARTITION, we present two strong intractability results that show the hardness of this problem even for very restricted instances.

We prove the first of these results, Theorem 4.3, by a reduction from the following NP-hard problem that we call k-PARTITION INTO INDEPENDENT SETS: given an undirected graph $G = (U, F)$, the task is to decide whether U can be partitioned into k independent sets of size $|U|/k$. This problem is NP-hard for $k = 3$, since it is equivalent to the special case of 3-COLORING where we require each color class to be of the same size. Moreover, k-PARTITION INTO INDEPENDENT SETS is also NP-hard in the case when $k = |U|/3$, by a reduction from the PARTITION INTO TRIANGLES problem (Garey and Johnson 1979) whose input is an undirected graph H and the task is to decide whether we can partition its vertex set into cliques of size 3. The construction resembles the one used in the proof of Theorem 3.2 but uses not only two, but a set of k dummies. Setting $k = 3$ in our reduction yields hardness for the setting $t = 4$, while setting $k = |U|/3$ yields our result for $b = 3$.

Theorem 4.3 ($\star$)**.** APPROXIMATE-CLONE PARTITION *is* NP*-complete, even if* $b = 3$ *or* $t = 4$*.*

The next result uses an intricate reduction from RX3C, relying on Proposition 2.3.

Theorem 4.4 ($\star$)**.** APPROXIMATE-CLONE PARTITION *is* NP*-hard even if the number of voters is constant* ($n = 17$)*.*

Contrasting Theorems 4.3 and 4.4, we now present an FPT algorithm for APPROXIMATE-CLONES PARTITION with parameter q, measuring the level of imperfection allowed. Hence, if q is a small constant, then we can find an approximate-clone partition efficiently, provided it exists.

Theorem 4.5. *There is an* FPT(q) *algorithm for* APPROXIMATE-CLONE PARTITION*.*

Proof. Consider an instance (E, b, q, t) of APPROXIMATE-CLONE PARTITION with election $E = (C, V)$. Our main idea is to convert it into an instance of the WEC, where the sets are intervals over $[m]$, for $m = |C|$, with possible "holes" on the first and last $2q$ entries. Such instances of WEC can be solved in FPT(q) time using Proposition 2.5.

Initializing the WEC Instance. Let $m = |C|$ be the number of candidates in E and let us fix an arbitrary voter $v \in V$. We rename the candidates so that $C = \{c_1, \ldots, c_m\}$ and

$$v : c_1 \succ_v c_2 \succ_v \cdots \succ_v c_m.$$

We form an instance of WEC with underlying set $[m]$, where each $i \in [m]$ corresponds to candidate $c_i \in C$. The sets included in the family $\mathcal{S}$ for WEC will correspond to (segments of) (q, b)-approximate clones, defined below.

Segments. Let Q be some (q, b)-approximate clone and let a and b be its top- and bottom-ranked members according to v. We say that a set P of candidates is *enclosed by* Q if for each $c \in P$ we have $a \succ_v c \succ_v b$. A set S of candidates is a *segment* if it can be partitioned into a (q, b)-approximate clone Q and a collection $P_1, \ldots, P_k$ of additional (q, b)-approximate clones such that each P_i is enclosed by Q. We refer to $Q, P_1, \ldots, P_k$ as an *implementation of* S, with Q being its *base*, and we let $1 + k$ be its *size*. The size of a segment S, denoted $\mathrm{size}(S)$, is the smallest among the sizes of its implementations.

Given a segment S, we define its *signature* as a length-m binary string such that for each $i \in [m]$, its i-th character is 1 if c_i belongs to S and it is 0 otherwise. Note that since a segment's base is a (q, b)-approximate clone, the segment's signature can have at most q 0s between its leftmost and rightmost 1.

An important observation is that an instance of APPROXIMATE-CLONE PARTITION is a yes-instance if and only if there is a partition of the candidate set C into segments $S_1, \ldots, S_{t'}$ such that none of them is enclosed by any other and $\sum_{i=1}^{t'} \mathrm{size}(S_i) \leq t$. We observe that each S_i in such a partition has the following property: If a 0 appears in its signature between the leftmost and rightmost 1, then it must be within $2q$ positions either from the leftmost 1 or from the rightmost 1. Indeed, otherwise one of the segments would be enclosed by another one (or would not be a segment at all, as any implementation for it would require the base not to be a (q, b)-approximate clone). We refer to the segments that satisfy this property as *valid*. Similarly, we say that signatures of valid segments are valid.

Adding Sets to the WEC Instance. For each valid segment S, we form a set for our WEC instance whose weight is $\mathrm{size}(S)$ and that includes exactly those elements $i \in [m]$ for which c_i belongs to S. We fix the threshold on the total weight of the desired solution for our WEC instance to be t. Note that there are at most $\mathcal{O}(\binom{4q}{q} m^2) = \mathcal{O}(2^{4q} m^2)$ (signatures of) valid segments: we have m^2 choices for the positions of the left- and rightmost 1 in the signature, and between them we have at most q 0s, each located at most $2q$ positions from the left- or the rightmost 1. Hence, it suffices to show that we can compute all valid segments.

Computing Valid Segments and Their Sizes. Given a valid signature, we show how to verify in FPT(q) time if it indeed corresponds to a segment and compute its size.

First, we let $S = \{e_1, \ldots, e_s\}$ be the set of candidates that correspond to the 1s in the signature (so if the first 1 is on position i in the preference order of v, then e_1 is c_i, and so on). Thus, S is the candidate set about which we want to decide whether it is a segment and, if so, establish its size. Suppose that $Q, P_1, \ldots, P_k$ is an implementation of S with the smallest size. We guess the size b' of Q; we have $b' \leq b$.

Next, we apply Corollary 3.4 for the candidate set S (recall $|S| \leq b + q$) and size b', and guess a set $Q' \subseteq S$ from its outputs, such that Q is a size-b' subset of Q' and, moreover, every b'-sized subset of Q' is a (q, b')-approximate clone.

We proceed to search for $P_1, \ldots, P_k$ that together with

$Q = Q' \setminus (P_1 \cup \cdots \cup P_k)$ have the following properties:

1. Each of $P_1, \ldots, P_k$ is a $(q, \min(q, b))$-approximate clone; indeed we are looking for a (q, b)-approximate clone partition, so each P_i needs to be (q, b)-approximate, and at most q members of S can belong to $P_1, \ldots, P_k$, so each of them must be (q, q)-approximate.
2. $P_1 \cup \ldots \cup P_k$ includes all the candidates in $S \setminus Q'$ as the segment S must include them, but they are not in Q.
3. $|Q| = b'$.

Due to the first property, for each $i \in [k]$ we see that there is a j such that P_i is a subset of $\{e_j, e_{j+1}, \ldots, e_{j+2q}\}$ of size at most q. Since for each such candidate set we can decide whether it forms a $(q, \min(q, b))$-approximate clone, it follows that we can compute in $\mathsf{FPT}(q)$ time a sequence $\mathcal{R} = R_1, R_2, \ldots$ of $(q, \min(q, b))$-approximate clones such that each P_i is equal to some R_ℓ from this list $\mathcal{R}$.

Furthermore, for each candidate $e \in S \setminus Q'$ there are at most 2^{2q} clones in the list $\mathcal{R}$ that include e. Hence, for each candidate $e \in S \setminus Q'$ we guess the clone $R_\ell \in \mathcal{R}$ that includes e and belongs to the implementation $Q, P_1, \ldots, P_k$ of S, ensuring that each of these guessed clones is either disjoint from the other ones or equal to one already guessed (as the same clone R_ℓ may include more than one candidate from $S \setminus Q'$). This way we obtain some first $P_1, \ldots, P_{k'}$ clones that satisfy the second of the above conditions, i.e., each $e \in S \setminus Q'$ is contained in some P_j, for $j \in [k']$.

Now we move on to compute the remaining $k - k'$ clones $P_{k'+1}, \ldots, P_k$, so that $|Q' \setminus (P_1 \cup \cdots \cup P_k)| = b'$. We let $Q'' = Q' \setminus (P_1 \cup \cdots \cup P_{k'})$ and $b'' = |Q''|$. We need $P_{k'+1}, \ldots, P_k$ to be disjoint subsets of Q'', each chosen from $\mathcal{R}$, whose total cardinality is exactly $b'' - b' \leq q$. Finding such a subset in $\mathsf{FPT}(q)$ time is possible by invoking the Partial Set Cover algorithm of Bläser (2003). His algorithm—based on color coding—finds a minimum-weight family of sets that covers at least a given number of elements; however, by only very minor adaptations of the dynamic programming part of his algorithm we can guarantee that the returned set family (if there is one) contains pairwise disjoint sets whose total size is exactly $b'' - b'$.

Altogether, after considering all the guesses, we either find that our signature corresponds to a segment and output its size (the smallest size of an implementation that we have found), or none of the guesses led to finding an implementation and the signature does not correspond to a segment.

Putting It All Together. Let us summarize our algorithm.

1. We consider all valid signatures, for each of them checking if it corresponds to a segment and computing its size. We collect all such segments in a WEC instance, where the segments' weights are their sizes.
2. We solve the WEC instance by applying Proposition 2.5, by observing that in the language of WEC, our segments are $2q$-approximate intervals.
3. We accept if the answer for our WEC instance is yes.

The correctness and $\mathsf{FPT}(q)$ running time follow from the preceding discussion. □

4.2 Independent Clones

Regarding independent-clone partitions, our first result rules out an efficient algorithm for finding such a partition even in the case when we have a constant number of voters and we are searching for clones of size at most three. It follows by a reduction from RX3C and relies on Proposition 2.3.

Theorem 4.6 (⋆). Independent-Clone Partition *is* NP*-complete, even if we require clones of size at most $b = 3$, each recognized by at least $w = 2$ voters, and the number of voters is a constant ($n = 14$).*

Yet, if the number of clones in the desired partition (that is, t) is constant, we can find this partition in polynomial-time by a simple brute-force approach.

Proposition 4.7. Independent-Clone Partition *has an* XP *algorithm with parameter t, the number of clones.*

Proof. We take all the preference orders of the voters in the input election (C, V) and merge them into a single list. Then, in this list, we guess at most t intervals; there are $\mathcal{O}((mn)^{2t})$ guesses to consider, where $m = |C|$ and $n = |V|$. For each group of intervals, we verify in polynomial time if it indeed corresponds to an independent-clone partition that meets the conditions of the input instance. □

Next, we show that Independent-Clones Partition is W[1]-hard for parameter t, which means that under standard complexity-theoretic assumptions we cannot improve the above algorithm to be an FPT one. We prove this by giving a parameterized reduction from the Perfect Code problem whose input is an undirected graph G and an integer k, and the task is to decide whether G admits a set S of k vertices such that each vertex of G is contained in the closed neighborhood of exactly one vertex from S. In fact, we need the multicolored version of this problem—where each vertex of G is assigned a color from $[k]$ and we require S to contain one vertex from each color class. NP-hardness of this variant follows from the NP-hardness of Perfect Code (Downey and Fellows 1995) by standard techniques.

Theorem 4.8 (⋆). Independent-Clones Partition *is* W[1]*-hard with parameter t, even if $w = 2$.*

4.3 Subelection Clones

While for approximate and independent clones we found efficient algorithms for computing partitions into clones of size at most two (recall Proposition 4.2), for subelection clones the same task is intractable.

Theorem 4.9 (⋆). Subelection-Clone Partition *is* NP*-complete, even if we require clones of size at most $b = 2$.*

Proof. To see that the problem is in NP, it suffices to guess a subelection and its perfect-clone partition, and verify that they meet the requirements of the given instance.

Next, we give a reduction from RX3C. Let $(X, \mathcal{S})$ be our input instance where $X = \{x_1, \ldots, x_{3k}\}$ is a set of $3k$ elements and $\mathcal{S} = \{S_1, \ldots, S_{3k}\}$ is a family of size-3 subsets of X. We form an election $E = (C, V)$ where

$$C = \{s_1, s_1^+, s_1^-, \ldots, s_{3k}, s_{3k}^+, s_{3k}^-\}$$

and V contains the following sets of voters:

1. There are $9k^2$ *type-1 voters*, each with preference order
$$s_1^+ \succ s_1 \succ s_1^- \succ s_2^+ \succ s_2 \succ s_2^- \succ \cdots$$
and $9k^2$ *type-2 voters*, each with preference order
$$s_1^- \succ s_1 \succ s_1^+ \succ s_2^- \succ s_2 \succ s_2^+ \succ \cdots$$
2. For each element $x_i \in X$ and each set S_j that contains x_i, we form a *coverage voter* $v_{i,j}$ whose preference order is the same as that of type-1 voters, except that we have the following modifications:
   (a) $v_{i,j}\colon s_j^- \succ s_j^+ \succ s_j$, and
   (b) for each $\ell \in [3k]$, if S_ℓ contains x_i but $\ell \neq j$, then $v_{i,j}\colon s_\ell^+ \succ s_\ell^- \succ s_\ell$.

   There are at most $9k^2$ coverage voters.

We set the upper bound on the clone sizes to be $b = 2$, the maximum number of clones to be $t = 6k$, and the number of voters in the subelection to be $w = 18k^2 + 3k$.

Let us assume that there is a subelection $E' = (C, V')$ with at least $w = 18k^2 + 3k$ voters that has a perfect-clone partition $\Pi = (C_1, \ldots, C_{t'})$ with $t' \leq t = 6k$ and $|C_i| \leq 2$ for each $i \in [t']$. By a simple counting argument, E' must include some type-1 and some type-2 voters and, hence, we can assume it includes all of them. Due to the preference orders of type-1 and type-2 voters, for each set S_j, either (i) Π contains $\{s_j, s_j^+\}$ and $\{s_j^-\}$, or (ii) Π contains $\{s_j, s_j^-\}$ and $\{s_j^+\}$, or (iii) Π contains each of $\{s_j\}, \{s_j^+\}, \{s_j^-\}$. However, this last option is impossible, as $|\Pi| \leq 6k$. Intuitively, if $\{s_j, s_j^+\} \in \Pi$ then S_j is included in a solution for the RX3C instance, and if $\{s_j, s_j^-\} \in \Pi$ then it is not.

Next, we observe that if E' includes some voter $v_{i,j}$ then, due to $v_{i,j}$'s preference order and the preceding paragraph, we must have $\{s_j, s_j^+\} \in \Pi$ and for every $\ell \in [3k] \setminus \{j\}$ such that $x_i \in S_\ell$ we must have $\{s_\ell, s_\ell^-\} \in \Pi$. This means that for each $i \in [3k]$, E' contains at most one coverage voter for x_i. However, since E' contains $18k^2 + 3k$ voters, it must contain one coverage voter for each $x_i \in X$. Thus, the sets S_ℓ for which $\{s_\ell, s_\ell^+\} \in \Pi$ must form an exact cover of X.

For the other direction, assume that $(X, \mathcal{S})$ is a yes-instance of the RX3C problem and that there are k sets from $\mathcal{S}$ whose union is X. For simplicity, let us assume that these are $S_1, \ldots, S_k$. We form a clone partition that contains clones $\{s_i, s_i^+\}$ and $\{s_i^-\}$ for each $i \in [k]$, and clones $\{s_\ell, s_\ell^-\}$ and $\{s_\ell^+\}$ for each $\ell \in [3k] \setminus [k]$. We form a subelection E' that includes all type-1 voters, all type-2 voters, and for each x_i we include a single coverage voter $v_{i,j}$ such that $j \in [k]$ and $x_i \in S_j$ (this is possible since $S_1, \ldots, S_k$ form an exact cover of X). This shows that we have a yes-instance of our SUBELECTION-CLONE PARTITION instance. □

On the positive side, we do have an FPT algorithm parameterized by the number n of voters. This strongly contrasts our results for the other type of clones, where this parameterization leads to intractability. We also find XP algorithms for the parameterizations by the size t of the partition and the number w of voters needed to recognize the clones.

Proposition 4.10. SUBELECTION-CLONE PARTITION *has an* FPT *algorithm for the parameterization by the number n of voters, an* XP *algorithm for parameterization by the number w of voters in the subelection, and an* XP *algorithm for the parameterization by the number t of clones.*

Proof. The $\mathsf{FPT}(n)$ algorithm guesses the voters that are included in the subelection and then runs the polynomial-time algorithm for PERFECT-CLONE PARTITION. The $\mathsf{XP}(w)$ algorithm proceeds similarly, by explicitly guessing w voters. The $\mathsf{XP}(t)$ algorithm first guesses a single voter to be included in the subelection, and then a partition of its preference order into at most t clones of appropriate sizes (i.e., it guesses a size-at-most-t perfect-clone partition consistent with this vote, where each clone contains at most b candidates). Finally, it checks if there are at least $w - 1$ additional voters that recognize all of these guessed clones. □

Corollary 4.11. SUBELECTION-CLONE PARTITION *can be solved in polynomial time for parameter $w = 2$.*

The XP algorithms from Proposition 4.10 cannot be improved to FPT ones as the problem is W[1]-hard even for $w + t$. The reduction is from the classic MULTICOLORED CLIQUE problem (Fellows et al. 2009; Pietrzak 2003).

Theorem 4.12 ($\star$)**.** SUBELECTION-CLONE PARTITION *is* W[1]*-hard with parameter $w + t$.*

5 Summary, Conclusions, and Future Work

We have introduced three natural types of imperfect clones and we have studied the complexity of (a) identifying them in elections and of (b) partitioning the candidates into clones of these types, with desired properties (such as the number of clones, their sizes, or their levels of imperfection). Overall, our problems are largely intractable, but we also found some polynomial-time, FPT, and XP algorithms for special cases. The main conclusion from our work is that if one were to solve our problems in practical settings, most likely one should use heuristic approaches, such as ILP solvers.

Our work can be extended in several ways. Foremost, while we tried to separate the different types of imperfect clones, it would be more realistic to study clones that can be both approximate and independent at the same time (i.e., only some voters would be required to recognize them, and even they could do so approximately). It would also be very interesting to perform experimental analysis of imperfect clones in elections. It is interesting to see how often they indeed appear, and for what levels of imperfection. Next, Cornaz, Galand, and Spanjaard (2012, 2013) described how one can use clones to generalize classic single-peaked (Black 1958) and single-crossing (Mirrlees 1971; Roberts 1977) domains, while maintaining good computational properties of resulting elections. It might be interesting to analyze if imperfect clones can be used in a similar way. Finally, it would be natural to consider approximate clones not only in the ordinal setting, but also in the approval one. In particular, this would open up the possibility of clone analysis in participatory budgeting elections, which often use approval data (Rey and Maly 2023).

Acknowledgments

This work is supported by the European Union (ERC, PRAGMA, 101002854). Views and opinions expressed are however those of the author only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. Grzegorz Lisowski acknowledges support by the European Union under the Horizon Europe project Perycles (Participatory Democracy that Scales). Ildikó Schlotter was supported by the Hungarian Academy of Sciences under its Momentum Programme (LP2021-2) and its János Bolyai Research Scholarship.

References

Berker, R.; Casacuberta, S.; Robinson, I.; Ong, C.; Conitzer, V.; and Elkind, E. 2025. From Independence of Clones to Composition Consistency: A Hierarchy of Barriers to Strategic Nomination. In *Proceedings of the 26th ACM Conference on Electronic Commerce (EC 2025)*, 1109.

Black, D. 1958. *The Theory of Committees and Elections*. Cambridge University Press.

Bläser, M. 2003. Computing Small Partial Coverings. *Information Processing Letters*, 85(6): 327–331.

Boehmer, N.; and Schaar, N. 2023. Collecting, Classifying, Analyzing, and Using Real-World Ranking Data. In *Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023)*, 1706–1715.

Cornaz, D.; Galand, L.; and Spanjaard, O. 2012. Bounded Single-Peaked Width and Proportional Representation. In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI 2012)*, 270–275.

Cornaz, D.; Galand, L.; and Spanjaard, O. 2013. Kemeny Elections with Bounded Single-Peaked or Single-Crossing Width. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, 76–82.

Cygan, M.; Fomin, F.; Kowalik, Ł.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M.; and Saurabh, S. 2015. *Parameterized Algorithms*. Springer.

Downey, R.; and Fellows, M. 1995. Fixed-Parameter Tractability and Completeness I: Basic Results. *SIAM Journal on Computing*, 24(4): 873–921.

Elkind, E.; Faliszewski, P.; and Slinko, A. 2011. Cloning in Elections: Finding the Possible Winners. *Journal of Artificial Intelligence Research*, 42: 529–573.

Elkind, E.; Faliszewski, P.; and Slinko, A. 2012. Clone Structures in Voters' Preferences. In *Proceedings of the 13th ACM Conference on Electronic Commerce (EC 2012)*, 496–513.

Faliszewski, P.; Janeczko, L.; Lisowski, G.; Pekárková, K.; and Schlotter, I. 2025. Identifying Imperfect Clones in Elections. Technical Report arXiv:2509.11261 [cs.GT], arXiv.org.

Fellows, M.; Hermelin, D.; Rosamond, F.; and Vialette, S. 2009. On the Parameterized Complexity of Multiple-Interval Graph Problems. *Theoretical Computer Science*, 410: 53–61.

Garey, M.; and Johnson, D. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company.

Gonzalez, T. 1985. Clustering to Minimize the Maximum Intercluster Distance. *Theoretical Computer Science*, 38: 293–306.

Janeczko, L.; Lang, J.; Lisowski, G.; and Szufa, S. 2024. Discovering Consistent Subelections. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024)*, 935–943.

Laffond, G.; Laine, J.; and Laslier, J. 1996. Composition Consistent Tournament Solutions and Social Choice Functions. *Social Choice and Welfare*, 13(1): 75–93.

Laslier, J. 1996. Rank-Based Choice Correspondencies. *Economics Letters*, 52(3): 279–286.

Laslier, J. 1997. *Tournament Solutions and Majority Voting*. Springer-Verlag.

Laslier, J. 2000. Aggregation of Preferences with a Variable Set of Alternatives. *Social Choice and Welfare*, 17(2): 269–282.

Mirrlees, J. 1971. An Exploration in the Theory of Optimal Income Taxation. *Review of Economic Studies*, 38: 175–208.

Niedermeier, R. 2006. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press.

Papadimitriou, C. 1994. *Computational Complexity*. Addison-Wesley.

Pietrzak, K. 2003. On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *Journal of Computer and System Sciences*, 67(4): 757–771.

Procaccia, A.; Schiffer, B.; and Zhang, S. 2024. Clone-Robust AI Alignment. In *Proceedings of the 42nd International Conference on Machine Learning (ICML 2025)*.

Rey, S.; and Maly, J. 2023. The (Computational) Social Choice Take on Indivisible Participatory Budgeting. Technical Report arXiv.2303.00621, arXiv.

Roberts, K. 1977. Voting Over Income Tax Schedules. *Journal of Public Economics*, 8(3): 329–340.

Tideman, T. 1987. Independence of Clones as a Criterion for Voting Rules. *Social Choice and Welfare*, 4(3): 185–206.