Deconstructing Intractability—A Case Study for Interval Constrained Coloring

Christian Komusiewicz*, Rolf Niedermeier, and Johannes Uhlmann**

Institut für Informatik, Friedrich-Schiller-Universität Jena,
Ernst-Abbe-Platz 2, D-07743 Jena, Germany.
{c.komus,rolf.niedermeier,johannes.uhlmann}@uni-jena.de

Abstract. The NP-hard Interval Constrained Coloring problem appears in the interpretation of experimental data in biochemistry dealing with protein fragments. Given a set of m integer intervals in the range 1 to n and a set of m associated multisets of colors (specifying for each interval the colors to be used for its elements), one asks whether there is a "consistent" coloring for all integer points from $\{1, \ldots, n\}$ that complies with the constraints specified by the color multisets. We initiate a study of Interval Constrained Coloring from the viewpoint of combinatorial algorithmics, trying to avoid polyhedral and randomized rounding methods as used in previous work. To this end, we employ the method of systematically deconstructing intractability. It is based on a thorough analysis of the known NP-hardness proof for Interval Constrained Coloring. In particular, we identify numerous parameters that naturally occur in the problem and strongly influence the problem's practical solvability. Thus, we present several positive (fixed-parameter) tractability results and, moreover, identify a large spectrum of combinatorial research challenges for Interval Constrained Coloring.

1 Introduction

Althaus et al. [1, 2] recently identified the Interval Constrained Coloring problem as an important combinatorial problem in the context of automated mass spectrometry and the determination of the tertiary structure of proteins. It builds the key to replace a manual interpretation of exchange data for peptic fragments with computer-assisted methods, see Althaus et al. [2] for more on the biochemical background and further motivation. The decision problem Interval Constrained Coloring (ICC) deals with matching color multisets with integer intervals and can be formalized as follows.[1] To this end, for two positive integers i, j with $i \leq j$, let $[i, j] := \{k \in \mathbb{N} \mid i \leq k \leq j\}$. Further, for $i \geq 1$ let $[i]$ denote the interval $[1, i]$.

* Supported by a PhD fellowship of the Carl Zeiss Foundation.
** Supported by the DFG, research project PABI, NI 369/7.

[1] Note that, compared with Althaus et al. [1, 2], we choose a somewhat different but equivalent formalization here; this problem definition turns out to be more suitable for our subsequent studies.

Input: A positive integer n, a set of m integer intervals $\mathcal{F} = \{F_1, \ldots, F_m\}$, all within $[n]$, a multiset of m multisets of colors $\mathcal{C} = \{C_1, \ldots, C_m\}$ over k different colors.
Question: Is there a coloring $c : [n] \to [k]$ such that for each interval $F_i \in \mathcal{F}$ it holds that $C_i = c(F_i)$?

Herein, $c(F_i)$ denotes the *multiset* of colors assigned by c to the integer points in the interval F_i. Concerning the biochemical background, the intervals correspond to (typically overlapping) fragments of a protein with n residues, and the k colors correspond to k different exchange rates that need to be assigned consistently to the n residues [1, 2]. The color multisets correspond to experimentally found bulk information that needs to be matched with the residues and can be interpreted as constraints that describe a set of valid colorings of the interval $[n]$ or, alternatively, a set of strings of length n over the "color alphabet". Note that from an applied point of view it is also important to investigate the corresponding optimization problems where one wants to maximize the number of requirements (that is, intervals that completely match with a given color multiset) that can be fulfilled [1]. However, in this paper we focus on analyzing the complexity of the decision problem.

Known results. To our knowledge, so far ICC has only been studied in the two papers by Althaus et al. [1, 2]. It has been shown to be NP-complete by a reduction from the Exact Cover problem [1]. In the more applied paper [2], besides first introducing and formalizing the problem, an algorithm based on integer linear programming and branch-and-bound was presented that enumerates all valid (fulfilling all constraints) color mappings c. In particular, it was shown that in the case of $k = 2$ colors a direct combinatorial algorithm leads to polynomial-time solvability; the computational complexity of the case $k = 3$ was left open. Finally, successful experiments with real-world instances with $n < 60$, $m \leq 50$, $k = 3$ and randomly generated instances with $n \leq 1000$, $m = n/2$, and $k = 3$ have been performed. In the more theoretical paper [1], besides the NP-completeness proof, the preceding work [2] has been continued by providing results concerning polynomial-time approximability. In particular, there is an algorithm producing a coloring where all requirements are matched within a mere additive error of one if the LP-relaxation of the presented integer program for ICC has a feasible solution. This algorithm is based on a sophisticated polyhedral approach combined with recent randomized rounding techniques.

Our contributions. This work proposes a fresh view on ICC and the development of exact algorithms for NP-hard combinatorial problems in general. The fundamental starting point here is to deconstruct proofs of NP-hardness in order to obtain new insights into the combinatorial structure of problems. More specifically, the point is to analyze how different parameters occurring in a problem contribute to its computational complexity. Having identified (some of) these parameters, the next step is to determine the complexity behavior in dependence on these parameters and combinations thereof. This is where parameterized algorithmics [5, 6, 8] comes into play. In case of ICC, there is an

enormous number of useful parameterizations, all naturally deduced from deconstructing the known NP-hardness proof. In this line, for instance, we can show a fixed-parameter tractability result with respect to the parameter "maximum interval length". Whereas we do not know whether the problem is fixed-parameter tractable with respect to the color parameter k alone, it is with respect to the combined parameter (n, k), that is, there is an algorithm with time complexity of the form $O^*((k-1)^n)$.[2] These algorithms are of practical interest when the corresponding parameter values are sufficiently small. For instance, note that all experiments of Althaus et al. [2] were performed having $k = 3$ and $n \leq 60$ for real-world instances. Indeed, in case of $k = 3$ we can further improve the running time to $O^*(1.89^n)$. In this spirit, in Section 3 we investigate a number of "single parameterizations", and in Section 4 we consider an even larger number of "combined parameterizations". Moreover, whereas ICC is NP-complete for "cutwidth" three [1], we present a combinatorial polynomial-time algorithm for cutwidth two. Tables 1 and 2 in Sections 3 and 4 survey the current state of the art and our new results concerning (combinatorial) algorithms that can efficiently solve ICC in case of favorable parameter constellations.

Due to the lack of space, several proofs are deferred to the long version of this paper.

2 Parameterization and Deconstruction of NP-Hardness

Parameterized algorithmics [5, 6, 8] aims at a multivariate complexity analysis of problems. The hope lies in accepting the seemingly inevitable combinatorial explosion for NP-hard problems, but to confine it to some parameter p. In this paper, p always is a positive integer or a vector of positive integers. A given parameterized problem (I, p) is *fixed-parameter tractable (FPT)* with respect to the parameter p if it can be solved within running time $f(p) \cdot \text{poly}(|I|)$ for some computable function f only depending on p.

A standard question of people unfamiliar with parameterized algorithmics is how to define respectively find "the" parameter for an NP-hard problem. There are the following (partly overlapping) "standard answers" to this question:
1. The standard parameterization typically refers to the size of the solution set of the underlying problem (whenever applicable).
2. A parameter describes a structural property of the input; for instance, the treewidth of a graph or the number of input strings.
3. Finding useful parameters to some extent is an "art" based on analyzing what typical real-world instances could look like.

Perhaps the most natural and constructive answer, however, is to look at the corresponding proof(s) of NP-hardness and what "parameter assumptions" they (do not) make use of. Indeed, this is nothing but what we mean by *deconstructing NP-hardness proofs for parameter identification*. In this work, we deconstruct the (only known) NP-hardness proof of ICC and gain a rich scenario of combinatorially and practically interesting structural parameterizations.

[2] The O^*-notation suppresses polynomial-time factors [9].

Let us now take a closer look at ICC. We first have to briefly review the known NP-hardness reduction from Exact Cover due to Althaus et al. [1]: The input of Exact Cover is a set $\mathcal{S}$ of subsets of a ground set $U := \{1, 2, \ldots, u\}$ and a positive integer t, and the question is whether there are t subsets from $\mathcal{S}$ such that every element from U is contained in exactly one such subset. Althaus et al.'s polynomial-time many-one reduction (using an approach by Chang et al. [4]) from Exact Cover to ICC works as follows.

1. The number of colors k is set to $s := |\mathcal{S}|$.
2. The interval range n is set to $(u + 1) \cdot s - t$.
3. For each element from U, there are exactly three corresponding integer intervals. Indeed, one can speak of three types of intervals, and all intervals of one type can be placed consecutively into one interval $[n]$ without overlap.
   (a) Type 1: Intervals of the form $[(i - 1)s + 1, is]$ for all $1 \leq i \leq u$.
   (b) Type 2: Intervals of the form $[is - t + 1, (i + 1)s - t]$ for all $1 \leq i \leq u$.
   (c) Type 3: Intervals of the form $[is - t - f_i + 1, is - t + 1]$ for all $1 \leq i \leq u$, where f_i denotes the number of occurrences of i in the sets of $\mathcal{S}$.
4. Every type-1 and every type-2 interval is assigned the color set $\{1, \ldots, k\}$. A type-3 interval corresponding to $i \in U$ is assigned the color set consisting of the colors associated with the subsets in $\mathcal{S}$ that contain i.

After having described the construction behind the NP-hardness proof, the deconstruction begins by making several observations about its properties:

1. The interval range n and the number m of intervals both are unbounded.
2. The number of colors k is s, hence unbounded, but all color multisets indeed are sets. That is, no interval shall be assigned the same color twice.
3. The maximum interval length is s, hence unbounded.
4. The maximum overlap between intervals is $\max\{t, s - t\}$, hence unbounded.
5. Only three different surrounding intervals $[n]$ are needed for comprising all intervals without overlap, hence the *cutwidth* of the constructed instance is bounded by three.

From the fifth observation we can conclude that there is no hope for fixed-parameter tractability with respect to the parameter "cutwidth" unless P=NP. On the positive side, we will show that ICC is polynomial-time solvable for cutwidth two. However, from the other four observations we directly obtain the following questions concerning a parameterized complexity analysis of ICC.

1. Is ICC fixed-parameter tractable with respect to the parameters n or m?
2. Is ICC fixed-parameter tractable with respect to k, or is it already NP-hard for constant k-values? Indeed, the complexity for the practically relevant case $k = 3$ is still unsettled. How does the parameter "maximum number of different colors per color multiset" influence the complexity? In the constructed instance this parameter is unbounded.
3. Is ICC fixed-parameter tractable with respect to the parameter "maximum interval length"?

4. Is ICC fixed-parameter tractable with respect to the parameter "maximum overlap between intervals"?

The central point underlying the above derived algorithmic questions is that whenever a quantity (that is, parameter) in an NP-hardness proof is unbounded (non-constant), then this evokes the quest to know what happens if this quantity is constant or considered to be small compared to the overall input size. Clearly, one way to answer is to provide a different proof of NP-hardness where this quantity is bounded. Otherwise, the main tool in answering such questions is parameterized algorithmics. Indeed, the story goes even further by combining different parameterizations. More specifically, it is, for instance, natural to ask whether ICC is fixed-parameter tractable when parameterized by both cutwidth and the number of colors k (the answer is open), or whether it is fixed-parameter tractable when parameterized by both n and k (the answer is "yes") and what the combinatorial explosion $f(n, k)$ then looks like. In this way, one ends up with an extremely diverse and fruitful ground to develop practically relevant combinatorial algorithms.

In the remainder of this paper, besides the already defined parameters n (range), m (number of intervals), and k (number of colors), we will consider the following parameters and combinations thereof:

- maximum interval length l;
- cutwidth $c := \max_{1 \leq i \leq n} |\{F \in \mathcal{F} : i \in F\}|$;
- maximum pairwise overlap between intervals $o := \max_{1 \leq i < j \leq m} |F_i \cap F_j|$;
- maximum number of different colors Δ in the color multisets.

Note that one of the integer linear programs devised by Althaus et al. [2] has $O(m \cdot k)$ variables. Using Lenstra's famous result [7] on the running time of integer linear programs with a fixed number of variables then implies that ICC is fixed-parameter tractable with respect to the (combined) parameter (m, k). Due to the huge combinatorial explosion in Lenstra's theorem, however, this result is of purely theoretical interest and more efficient combinatorial algorithms are of big interest.

In the next two sections, we present several fixed-parameter tractability results with respect to the above parameters (Section 3) and combinations of each time two of them (Section 4).

Let us spot some challenges for future research concerning the multivariate complexity analysis of ICC. The complexity behavior with respect to the combined parameter (c, k) is widely open. A breakthrough would be to show the tractability with respect to k—note that we have intractability with respect to c. Basically along the same lines as k, also Δ gives an interesting parameterization with almost no results so far. The parameter m also seems of particular interest. The fixed-parameter tractability with respect to m is completely open and with respect to the combined parameter (m, k) the running time needs improvement.

We close this section with some simple observations about a helpful "normal form" that one may assume without loss of generality for all ICC input instances. More precisely, based on simple and efficient preprocessing rules, one can perform a data reduction that yields this normal form.

Table 1. Complexity of ICC for 1-dimensional parameterizations. Herein, "P" means that the problem is polynomial-time solvable, "NPc" means that the problem is NP-complete, and "?" means that the complexity is unknown. For fixed-parameter algorithms, we only give the function of the exponential term, omitting polynomial factors. The results for $k = 2$ and $c = 3$ are due to Althaus et al. [1, 2], the rest is new.

Parameter	k	Δ	l	c	m	n	o
Complexity	$k = 2$: P $k \geq 3$: ?	$\Delta \geq 2$: ?	$l!$	$c = 2$: P $c = 3$: NPc	?	$n!$	$o = 1$: P $o \geq 2$: ?

Proposition 1. (Normal form for ICC)
In $O(lmn)$ time, one can transform every ICC *instance into an equivalent one such that*

1. *at every position $i \in [n]$, there is at most one interval starting at i and at most one interval ending at i, and*
2. *if the maximum interval length is l, then every position $i \in [n]$ is contained in at most $2l$ intervals.*

3 Single Parameters

In Section 2, we identified various parameters as meaningful "combinatorial handles" to better assess the computational complexity of ICC. Concerning cutwidth c, whereas $c = 3$ is known to be NP-complete [1], here we show that $c = 2$ is polynomial-time solvable. Obviously, $l \leq n$, so the fixed-parameter tractability with respect to l (as we will prove subsequently) implies the fixed-parameter tractability with respect to n. Table 1 surveys known and new results with respect to single parameters.

Theorem 1. ICC *can be solved in $O(l! \cdot lmn)$ time.*

Proof. We present a dynamic programming algorithm. We use the following notation. First, let A denote the set of intervals contained in some other intervals, that is, $A := \{F \in \mathcal{F} \mid \exists_{F' \in \mathcal{F}} : F \subseteq F'\}$, and $B := \mathcal{F} \setminus A$. Let $\mathcal{K} = \{1, \ldots, k\}$ denote the set of all colors. We say that a coloring c' *satisfies* an input interval $F_i \in \mathcal{F}$ if $c'(F_i) = C_i$. For an interval $[s, t]$, a coloring is represented by a vector in $\mathcal{K}^{t-s+1}$. For an input interval $F_i \in \mathcal{F}$, the set K_i of all satisfying colorings is given by $K_i := \{c' \in \mathcal{K}^{|F_i|} \mid c' \text{ satisfies } C_i\}$. In the worst case that every color occurs at most once in the multiset C_i, there are $|C_i|!$ satisfying colorings of an input interval F_i. In the following, we assume that the intervals in B are ordered in increasing order of their start points (and, hence, also in increasing order of their endpoints). Let $B = \{B_1, \ldots, B_{m'}\}$ and $B_j = [s_j, t_j]$ for all $1 \leq j \leq m'$. The intervals in B cover $[n]$, that is, $\bigcup_{j=1}^{m'} B_j = [n]$. For every B_j, the algorithm maintains a table T_j with an entry for every satisfying

coloring of B_j. More specifically, for every coloring $c' = (c'_1, \ldots, c'_{|B_j|}) \in K_j$ we set $T_j(c') = \text{true}$ iff there exists a coloring $c'' = (c''_1, \ldots, c''_{t_j}) \in \mathcal{K}^{t_j}$ of the interval $[t_j]$ with $(c''_{s_j}, \ldots, c''_{t_j}) = c'$ such that c'' satisfies each interval $F \in \mathcal{F}$ with $F \subseteq [t_j]$.

For $j = 1$ and for every $c' \in K_1$, this is achieved by setting $T_1(c') := \text{true}$ iff c' satisfies every interval $[s,t] \in \mathcal{F}$ with $[s,t] \subseteq [s_1, t_1]$.

We say that a coloring $c' = (c'_1, \ldots, c'_{|B_j|}) \in K_j$ for B_j is consistent with a coloring $c'' = (c''_1, \ldots, c''_{|B_{j-1}|}) \in K_{j-1}$ for B_{j-1} if c' and c'' agree in $B_{j-1} \cap B_j$, that is, $(c''_{s_j - s_{j-1}+1}, \ldots, c''_{|B_{j-1}|}) = (c'_1, \ldots, c_{t_{j-1}-s_j+1})$. We write $c'|c''$ to denote that c' is consistent with c''.

For $j = 2, \ldots, n$ and for every $c' = (c'_1, \ldots, c'_{|B_j|}) \in K_j$, we set

$$T_j(c') = \text{true} \iff c' \text{ satisfies all } F \in \mathcal{F} \text{ with } F \subseteq B_j \text{ and } \exists_{c'' \in K_{j-1},\, c'|c''} : T_{j-1}(c'') = \text{true}.$$

The correctness can be seen as follows. The "$\Rightarrow$"-direction follows directly by definition. For the "$\Leftarrow$"-direction, observe the following. A coloring of $[t_j]$, composed of a coloring of $[t_{j-1}]$ satisfying all $F \in \mathcal{F}$ with $F \subseteq [t_{j-1}]$ and a coloring c' of $[s_j, t_j]$ satisfying all $F \in \mathcal{F}$ with $F \subseteq [s_j, t_j]$, satisfies all $F \in \mathcal{F}$ with $F \subseteq [t_j]$; clearly, all $F \in \mathcal{F}$ with $F \subseteq [t_{j-1}]$ are satisfied. Moreover, all other $F \in \mathcal{F}$ with $F \subseteq [t_j]$ are satisfied since for every fragment $[s,t] \in \mathcal{F}$ with $t_{j-1} < t \leq t_j$ it holds that $[s,t] \subseteq [s_j, t_j]$.

As to the running time, there are at most $|B_j|!$ satisfying colorings of B_j; at most one for every permutation of the associated color multiset. Hence, one has to consider at most $l!$ colorings for every B_j. For every $j = 1, \ldots, m' - 1$, the algorithm works as follows. When building the table T_j for every $c' = (c'_1, \ldots, c'_{|B_j|}) \in K_j$, the algorithm computes an auxiliary table Q_j with an entry $Q_j(c'_r, c'_{r+1}, \ldots, c'_{|B_j|})$, where $r := s_{j+1} - s_j + 1$, indicating whether $T_j(c') = \text{true}$. Herein, in order to ensure that the size of Q_j does not exceed $l!$ and to allow fast access to its elements, table Q_j can for example be realized as an array of size $|B_j|!$ where the entry for $c_s = (c'_r, c'_{r+1}, \ldots, c'_{|B_j|})$ is stored at the position corresponding to the number of the lexicographically smallest permutation of C_j with "prefix" c_s. Then, to check whether $\exists_{c'' \in K_{j-1},\, c'|c''} : T_{j-1}(c'') = \text{true}$ for a $c' = (c'_1, \ldots, c'_{|B_j|}) \in K_j$, the algorithm can check whether $Q_{j-1}(c'_1, \ldots, c'_{|B_j|}) = \text{true}$ in $O(l)$ time. Hence, for every position $1 \leq j \leq m'$, it needs at most $O(l! \cdot (l + lm))$ time, where the factor lm is due to checking whether c' satisfies all $F \in \mathcal{F}$ with $F \subseteq [s_j, t_j]$. In summary, the total running time is $O(l! \cdot lmn)$ since $m' \leq n$. □

Next, we show that ICC is solvable in $O(n^2)$ time for $c = 2$. This contrasts the case $c = 3$ shown to be NP-complete [1]. Our algorithm is based on four data reduction rules that are executable in polynomial time. The application of these rules either leads to a much simplified instance that can be colored without violating any interval constraints or shows that the instance is a no-instance.

Reduction Rule 1 *For any two intervals F_i and F_j,*

– *if* $|F_i \cap F_j| = |C_i \cap C_j|$, *then set* $c(F_i \cap F_j) = C_i \cap C_j$;
– *if* $|F_i \cap F_j| > |C_i \cap C_j|$, *then return "No".*

Rule 1 is obviously correct: if two intervals "share" more positions than colors, then there is no coloring that satisfies both intervals, and if the number of shared positions is equal to the number of shared colors, then we have to color the overlapping intervals exactly with these colors.

Note that when we set $c(i) = c_x$ for some position i, we can simplify the instance as follows. For all $F_j = [s,t]$ with $s \leq i \leq t$, we set $C_j := C_j \setminus \{c_x\}$ and $t := t - 1$. For all $F_j = [s,t]$ with $i < s$, we set $s := s - 1$ and $t := t - 1$. "Empty" intervals F_j with $C_j = \emptyset$ are removed from the input. After Rule 1 and this subsequent reduction of the instance, we can assume that no interval is completely contained in any other interval.

In the following, assume that the intervals are ordered with respect to their startpoints, that is, for $F_i = [s_i, t_i]$ and $F_j = [s_j, t_j]$ with $i < j$ we have $s_i < s_j$. Let t be the endpoint of the first interval $F_j \neq F_1$ that overlaps only with one other interval. Clearly, we can color $[t]$ independently from $[t+1, n]$. Together with Rule 1, and the fact that $c = 2$, we can thus assume that all intervals except for F_1 and F_m overlap with exactly two other intervals. Hence, we can partition each interval F_j, $1 < j < m$, into at most three subintervals: the first subinterval overlaps with F_{j-1}, the second (possibly empty) subinterval does not overlap with any other interval, and the third subinterval overlaps with F_{j+1}. The following notation describes this structural property. For an interval F_j, $1 < j < m$, define $F_j^1 := F_j \cap F_{j-1}$, $F_j^3 := F_{j+1}^1$, and let $F_j^2 := F_j \setminus (F_j^1 \cup F_j^3)$. For a coloring c' of all intervals, let $C_j^1 := c'(F_j^1)$. Define C_j^2 and C_j^3 accordingly. For F_1, define $F_1^3 := F_2^1$, and $F_1^2 := F_1 \setminus F_1^3$; for F_m define $F_m^1 := F_m \cap F_{m-1}$ and $F_m^2 := F_m \setminus F_m^1$; C_1^3, C_1^2, C_m^1, and C_m^2 are defined analogously. Whether a coloring violates an interval F_j only depends on the sets C_j^1,C_j^2, and C_j^3. Hence, when we know that a color c_x must belong to some C_j^l, $1 \leq l \leq 3$, then we can color an arbitrary $i \in F_j^l$ with c_x. Finally, let $\mathrm{occ}(x, C)$ denote the multiplicity of an element x in a multiset C.

The next rule reduces intervals F_j that have no "private" middle interval F_j^2 but more elements of a color c_x than the previous interval.

Reduction Rule 2 *For any interval* F_j, *if* $F_j^2 = \emptyset$ *and there is a color* c_x *such that* $\mathrm{occ}(c_x, C_{j-1}) < \mathrm{occ}(c_x, C_j)$, *then set* $c(i) = c_x$ *for some arbitrary* $i \in F_j^3$.

The rule is obviously correct. After its application, for every interval F_j with $F_j^2 = \emptyset$, we have $|F_{j-1}| > |F_j|$. Next, we reduce triples of intervals F_{j-1}, F_j, F_{j+1} that have identical color multisets in case $F_j^2 = \emptyset$.

Reduction Rule 3 *For intervals* F_{j-1}, F_j, *and* F_{j+1} *such that* $C_{j-1} = C_j = C_{j+1}$ *and* $F_j^2 = \emptyset$, *remove* F_{j-1} *and* F_j *from the input and for all intervals* $F_{j+l} = [s,t]$ *with* $l \geq 1$ *set* $F_{j+l} = [s', t']$, *where* $s' := s - |F_j|$ *and* $t' := t - |F_j|$.

The correctness proof for Rule 3 is omitted.

The following is our final data reduction rule.

Reduction Rule 4 *Let I be an instance that is reduced with respect to Rules 1, 2, and 3, and let F_j be the first interval of I such that there is a color c_x with $\mathrm{occ}(c_x, C_j) > \mathrm{occ}(c_x, C_{j+1})$. Do the following:*

- *if $j = 1$, then set $c(i) = c_x$ for some $i \in F_1^2$;*
- *if $j > 1$ and $c_x \notin C_{j-1}$, then set $c(i) = c_x$ for some $k \in F_j^2$ in case $F_j^2 \neq \emptyset$ and otherwise return "No";*
- *if $j > 1$ and $c_x \in C_{j-1}$, then set $c(i) = c_x$ for some $i \in F_j^1$.*

Lemma 1. *Rule 4 is correct.*

Proof. Let I be an instance, reduced with respect to Rules 1, 2, and 3, to which Rule 4 is applied, and let I' be the resulting instance. We only show that if I is a yes-instance, then I' is a yes-instance, since the reverse direction trivially holds.

If $j = 1$, this is easy to see: since c_x occurs more often in F_1 than in F_2 one of the positions in $F_1 \setminus F_2$ must be colored with c_x.

If $j > 1$ and $c_x \notin C_{j-1}$, then it is clear that one of the positions in F_j^2 must be colored with c_x. We either perform this forced coloring or return "No" if this is not possible.

Finally, if $j > 1$ and $c_x \in C_{j-1}$, the situation is more complicated. Let c' be a coloring that fulfills the interval constraints of I, we call such a coloring *proper*. If there is a position $i \in F_j^1$ such that $c'(i) = c_x$, then the claim obviously holds. Otherwise, we show that we can transform c' into an alternative coloring c'' that is proper and there is an $i \in F_j^1$ such that $c''(i) = c_x$. Whether coloring c'' is proper depends only on the multisets C_l^1, C_l^2, and C_l^3, $1 \leq l \leq m$, that are defined by the coloring function c'. Hence, we describe the transformation applied to c' with respect to these multisets. Note that we do not modify the sets C_l^y for any $l > j$.

We face the following situation: $c_x \notin C_j^1$, but since c' is a coloring that does not violate any interval constraints and by the precondition of Rule 2, $c_x \in C_j^2$. By the precondition of Rule 4, we have $C_1 \subseteq C_2 \subseteq \ldots \subseteq C_j$. We show that we can always find a series of exchange operations such that the resulting coloring is proper and $c_x \in C_j^1$. We perform a case distinction. Due to the lack of space, we show only some cases, the other cases are similar, albeit more complicated.
Case 1: $F_{j-1}^2 \neq \emptyset$. There are three subcases of this case.
Case 1.1: $c_x \in C_{j-1}^2$. In this case, we exchange $c_x \in C_{j-1}^2$ and some arbitrary $c_l \in C_j^1$. Furthermore, we remove c_x from C_j^2 and add c_l to C_j^2. The exchange is shown in Fig. 1a; the resulting coloring is clearly proper and $c_x \in C_j^1$.
Case 1.2: $c_x \in C_{j-1}^1$ and $F_{j-2}^2 \neq \emptyset$. Clearly, C_{j-2} must be involved in the exchange. We choose an arbitrary element $c_l \in C_{j-2}^2$. Since $C_{j-2} \subseteq C_{j-1}$, we also have $c_l \in C_{j-1} \setminus C_{j-2}$. We distinguish two subcases.
Case 1.2.1: $c_l \in C_{j-1}^3$. We perform a *direct* exchange of c_l and c_x between C_{j-2}^2 and C_{j-1}^1 and also between C_j^1 and C_j^3. The exchange is shown in Fig. 1b; the resulting coloring is clearly proper and $c_x \in C_j^1$.

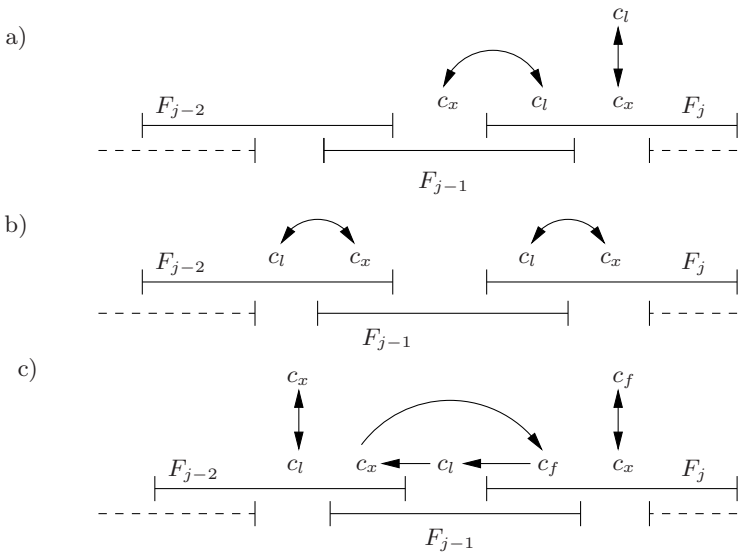

Fig. 1. Exchange operations used in the proof of Lemma 1.

Case 1.2.2: $c_l \in C^2_{j-1}$**.** We remove c_l from C^2_{j-2} and add c_x to C^2_{j-2}. Furthermore, we perform a *circular* exchange between C^1_{j-1}, C^2_{j-1}, and C^3_{j-1}: move c_x from C^1_{j-1} to C^3_{j-1}, move an arbitrary element c_f from C^3_{j-1} to C^2_{j-1}, and move c_l from C^2_{j-1} to C^1_{j-1}. Finally, we remove c_x from C^2_j and add c_f to C^2_j. The exchange is shown in Fig. 1c; the resulting coloring is clearly proper and $c_x \in C^1_j$.

The correctness of the final two cases is deferred to a long version of this paper.

Case 1.3: $c_x \in C^1_{j-1}$ **and** $F^2_{j-2} = \emptyset$**.**

Case 2: $F^2_{j-1} = \emptyset$**.**

In all cases, we can construct an alternative coloring that is proper and $c_x \in C^1_j$, which means that we can assume that if I is yes-instance, then there is some $i \in C^1_j$ such that $c'(i) = c_x$. In summary, this shows that I is a yes-instance iff I' is a yes-instance. □

With these four reduction rules at hand, we can describe a simple quadratic-time algorithm for ICC with cutwidth two.

Theorem 2. ICC *can be solved in* $O(n^2)$ *time when the input has cutwidth two.*

Proof. The algorithm starts with exhaustively applying Rules 1 to 4. Note that the rules have to be applied in the correct order, that is, after each reduction step, we always check first whether Rule 1 can be applied, then whether Rule 2 can be applied, and so on. The rules either return "No" or we obtain an instance that is reduced with respect to all data reduction rules. In such an instance we have $C_1 \subseteq C_2 \subseteq \ldots \subseteq C_m$. Otherwise, Rule 4 would apply, because there would

be some $F_i \in \mathcal{F}$ and a color c_x such that $\mathrm{occ}(c_x, C_i) > \mathrm{occ}(c_x, C_i + 1)$. This instance can be easily colored as follows. For the first interval F_1, we choose an arbitrary coloring that does not violate C_1. Clearly, this also does not violate C_2 since $C_1 \subseteq C_2$. Then we remove the colored parts from the input, adjust the color multisets accordingly, and choose an arbitrary coloring that does not violate C_2. Clearly, this does not violate C_3, since $C_2 \subseteq C_3$. After this, we again reduce the colored parts and continue with coloring F_3. This is repeated until all positions are colored and clearly produces a coloring that does not violate any interval constraints. This proves the correctness of the algorithm.

For the running time of the algorithm consider the following. First, since the input has cutwidth two, we have $m = O(n)$. For each data reduction rule, checking whether it can be applied and the application itself can be performed in $O(n)$ time. Furthermore, the application of any of the reduction rules removes at least one position from the interval $[n]$. Hence, each rule can be applied at most n times. Together with the $O(n)$ time that is clearly sufficient for coloring any instance reduced with respect to the reduction rules, this leads to a total running time of $O(n^2)$. □

Using the previous algorithm, we also obtain polynomial-time solvability in case the maximum overlap o between fragments is at most one. This follows from the observation that after achieving the normal form of the instance, each instance with overlap at most one also has cutwidth at most two, which can be seen as follows. Suppose an instance that has the normal form has overlap one and cutwidth at least three. Then there must be an interval F_i that overlaps with two other intervals F_j, F_l at some position x. Since at each position at most one interval starts, at most one of F_j and F_l, say F_j, starts at position x. This, however, means that F_l starts at some position $u < x$ and hence it has overlap at least two with F_i, leading to a contradiction.

Corollary 1. ICC *can be solved in* $O(n^2)$ *time when the input has overlap one.*

4 Combined Parameters

In the following, as already indicated in Section 2, we turn to the study of some relevant pairs of single parameters which form a "combined parameter". Table 2 summarizes our current knowledge about combined parameterizations of ICC—there are many questions left open. Here, we study the three different combined parameters (l, k), (l, c), and (n, k) that seem to be of immediate practical interest and all allow for fixed-parameter tractability results. The proof of Theorem 3 is somewhat similar to the proof of Theorem 1 and therefore omitted.

Theorem 3. ICC *can be solved in* $O(k^l \cdot (k + lm)n)$ *time.*

Theorem 4. ICC *can be solved in* $(c + 1)^l \cdot \mathrm{poly}(n, m)$ *time.*

Proof. We present a dynamic programming algorithm. We use the following notation. For every position i, $1 \leq i \leq n$, let $\mathcal{F}_i = \{F_{i_1}, \ldots, F_{i_{n_i}}\}$ denote the

Table 2. Complexity of ICC for combined parameters. We only give the function of the exponential term, omitting polynomial factors. Herein, $(k,*)$ and $(k,*,*)$ refer to combined parameters that feature k and one or two additional parameters, $(l,*)$ refers to combined parameters that feature l and one additional parameter. Note that for $k=3$, we achieve an improvement from 2^n to 1.89^n. The result for parameter (k,m) is due to Althaus et al. [2], the rest is new.

Parameter	Running times
$(k,*)$	k^l, $(k-1)^n$, $f(k,m)$ (ILP)
$(k,*,*)$	$l^{c\cdot(k-1)}$, $n^{c\cdot(k-1)}$
$(l,*)$	Δ^l, $(c+1)^l$

input intervals containing i. Further, let $\mathcal{C}_i = \{C_{i_1}, \ldots, C_{i_{n_i}}\}$ denote the color multisets associated with the intervals in $\mathcal{F}_i$, where C_{i_j} is the color multiset associated with F_{i_j}, $1 \leq j \leq n_i$. Note that $n_i \leq c$. Further, let $F_{i_j} = [s_{i_j}, t_{i_j}]$ for all $1 \leq j \leq n_i$. By $\mathcal{K} = \{1, \ldots, k\}$ we refer to the set of all colors. Further, a tuple $(M_1, \ldots, M_q)$ of (multi)sets is called a *chain* if there exists a permutation π of $\{1, \ldots, q\}$ such that $M_{\pi(1)} \subseteq M_{\pi(2)} \subseteq \ldots \subseteq M_{\pi(q)}$.

For every position i, the algorithm maintains a table T_i with a Boolean entry for every possible tuple of color multisets $(A_1, \ldots, A_{n_i})$ with $A_j \subseteq C_{i_j}$ and $|A_j| = i - s_{i_j} + 1$ for all $1 \leq j \leq n_i$. More specifically, $T_i(A_1, \ldots, A_{n_i})$ is true iff there exists a coloring $c' : [i] \to \mathcal{K}$ such that $c'([s_{i_j}, i]) = A_j$ for all $1 \leq j \leq n_i$ and for every $F_l \in \mathcal{F}$ with $F_l \subseteq [i]$ it holds that $c'(F_l) = C_l$. Note that an instance is a yes-instance iff there is a true entry in T_n. The goal of the dynamic programming is to compute the tables T_i to fulfill this definition.

According to Proposition 1, at each position in $[n]$ there starts at most one input interval and ends at most one input interval. Hence, there is exactly one interval in $\mathcal{F}_1$. Let $\mathcal{F}_1 = \{F\}$ and let C be the color multiset associated with F. We set $T_1(\{c'\}) = \text{true}$ for every $c' \in C$.

For every position i, $2 \leq i \leq n$, there is at most one interval in $\mathcal{F}_{i-1} \setminus \mathcal{F}_i$ and at most one in $\mathcal{F}_i \setminus \mathcal{F}_{i-1}$. Thus, assume that $\mathcal{F}_{i-1} = \{F', F_1, \ldots, F_q\}$ and $\mathcal{F}_i = \{F_1, \ldots, F_q, F''\}$, that is, $\mathcal{F}_{i-1} \cap \mathcal{F}_i = \{F_1, \ldots, F_q\}$ (if $\mathcal{F}_{i-1} \setminus \mathcal{F}_i = \emptyset$ or $\mathcal{F}_i \setminus \mathcal{F}_{i-1} = \emptyset$, then skip F' or F'' in the following formulas). Let $F_j = [s_j, t_j]$ for all $1 \leq j \leq q$.

For every tuple $(A_1, \ldots, A_q, A'')$ that forms a chain and fulfills $A_j \subseteq F_j$, $|A_j| = i - s_j + 1$ for $1 \leq j \leq q$ and $A'' \subseteq F''$, $|A''| = 1$, set

$$T_i(A_1, \ldots, A_q, A'') = \text{true} \iff \exists_{x \in (\bigcap_{j=1}^{q} A_j) \cap A''} : T_{i-1}(F', A_1 \setminus \{x\}, \ldots, A_q \setminus \{x\}). \quad (I)$$

The correctness of this recursion can be seen as follows. On the one hand, if $T_i(A_1, A_2, \ldots, A_q, A'') = \text{true}$, that is, if there exists a coloring $c' : [i] \to \mathcal{K}$ fulfilling the above conditions, then clearly c' restricted to $[i-1]$ fulfills the above properties for $i-1$ and $F', A_1 \setminus \{c'(i)\}, \ldots, A_q \setminus \{c'(i)\}$.

On the other hand, if $\exists_{x \in (\bigcap_{j=1}^{q} A_j) \cap A''} : T_{i-1}(A', A_1 \backslash \{x\}, \ldots, A_q \backslash \{x\}) = \text{true}$, that is, if there exists a coloring $c'' : [i-1] \to \mathcal{K}$ fulfilling the above properties for $i-1$ and $A', A_1 \backslash \{x\}, \ldots, A_q \backslash \{x\}$, then the extension c' of c'' with $c'(j) = c''(j)$ for $1 \leq j < i$ and $c'(i) = x$ fulfills the above properties for i and $A_1, \ldots, A_q, A''$.

Finally, note that we only consider tuples $(A_1, \ldots, A_q, A'')$ that form chains. This is correct, since for a coloring $c' : [i] \to \mathcal{K}$ it clearly holds that the tuple $(c'([s_1, i]), \ldots, c'([s_j, i]), \{c(i)\})$ forms a chain.

In accordance with the equivalence (I), the algorithm computes the tables T_i for increasing values of i (starting with $i = 2$). Finally, it outputs "Yes" if T_n contains a true entry and "No", otherwise.

As to the running time, for every position there are at most $(c+1)^l$ tuples of color multisets $(A_1, \ldots, A_{n_i})$ with $A_j \subseteq C_{i_j}$ and $|A_j| = i - s_{i_j} + 1$, $1 \leq j \leq n_i$, that form a chain. This can be seen as follows. Let F_l denote the interval in $\mathcal{F}_i$ with the smallest starting point. Clearly, a tuple of color multisets $(A_1, \ldots, A_{n_i})$ that forms a chain corresponds to a partition of C_j into $(c+1)$ subsets. Since, for every color in F_j there are $(c+1)$ choices, there are at most $(c+1)^l$ such partitions. □

For a multiset M that contains k different colors and for an integer $q \geq 1$ there are at most q^{k-1} size-q submultisets of M. This is true since if we have chosen the occurrence number of the first $k-1$ colors in a size-q subset (there are at most q^{k-1} choices), then the occurrence number of the kth color is fixed. With this observation, the running time of the algorithm presented in the proof of Theorem 4 can also be bounded by $O^*(l^{c \cdot (k-1)})$.

Corollary 2. ICC *can be solved in* $l^{c \cdot (k-1)} \cdot \text{poly}(n, m)$ *time.*

Trivially, we can solve any instance in $O(k^n)$ time by trying all k colors for all n positions. Now, we use the fact that for two colors the problem is polynomial-time solvable [2]. Hence, we need to "guess" only $k-2$ colors and the positions that have one of the two remaining colors. For these positions, we then use the polynomial-time algorithm for ICC with two colors, giving the following result.

Theorem 5. ICC *can be solved in* $O((k-1)^n \cdot g(n, m))$ *time, where* $g(n, m)$ *is the time needed to solve* ICC *for* $k = 2$.

For the practically relevant case where $k = 3$ [2], we can achieve a speed-up by the following simple observation: At least one of the colors appears at most on $n/3$ positions.

Theorem 6. *For* $k = 3$, ICC *can be solved in* $O(1.89^n \cdot g(n, m))$ *time, where* $g(n, m)$ *is the time needed to solve* ICC *for* $k = 2$.

Beigel and Eppstein [3] gave a thorough study of exact exponential-time algorithms for the NP-complete 3-Coloring problem. It is tempting to investigate whether some of their tricks can be applied to ICC with three colors; in particular, a simple randomized strategy presented by Beigel and Eppstein might be promising.

5 Conclusion

Conceptually, we presented a systematic development of the method of "deconstructing intractability". We exhibited this approach using the NP-complete ICC problem as a particularly fertile application case. Through deconstruction and using methods of parameterized algorithmics, we started a diverse multivariate complexity analysis of ICC. Refer to Tables 1 and 2 in Sections 3 and 4 for an overview and numerous challenges for future research. There remain many challenges for future work: Even combinations of three or more parameters may be relevant. Besides that, already for pairs of two single parameters there are several qualitatively different fixed-parameter tractability results one can strive for and which typically are independent from each other. For instance, for a combined parameter (p_1, p_2) combinatorial explosions such as $p_1^{p_2}$, $p_2^{p_1}$, $2^{p_1 \cdot p_2}$ etc. all can be useful for solving specific real-world instances. Finally, we focussed attention on the decision version, but the investigations should clearly be extended to the optimization variants. Summarizing, the research challenges offered by ICC and, more generally, the deconstructive approach to intractability, seem to be (almost) inexhaustible.

Acknowledgment. We thank Michael R. Fellows for early discussions about the deconstructive approach to NP-hard problems and Nadja Betzler for pointing us to the ICC problem.

References

[1] E. Althaus, S. Canzar, K. Elbassioni, A. Karrenbauer, and J. Mestre. Approximating the interval constrained coloring problem. In *Proceedings of the 11th Scandinavian Workshop on Algorithm Theory (SWAT '08)*, volume 5124 of *LNCS*, pages 210–221. Springer, 2008.

[2] E. Althaus, S. Canzar, M. R. Emmett, A. Karrenbauer, A. G. Marshall, A. Meyer-Baese, and H. Zhang. Computing H/D-exchange speeds of single residues from data of peptic fragments. In *Proceedings of the 23rd ACM Symposium on Applied Computing (SAC '08)*, pages 1273–1277. ACM, 2008.

[3] R. Beigel and D. Eppstein. 3-coloring in time $O(1.3289^n)$. *Journal of Algorithms*, 54(2):168–204, 2005.

[4] J. Chang, T. Erlebach, R. Gailis, and S. Khuller. Broadcast scheduling: Algorithms and complexity. In *Proceedings of the 19th ACM-SIAM Symposium on Discrete Algorithms (SODA '08)*, pages 473–482. ACM-SIAM, 2008.

[5] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

[6] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006.

[7] H. W. Lenstra. Integer programming with a fixed number of variables. *Mathematics of Operations*, 8:538–548, 1983.

[8] R. Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Number 31 in Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, 2006.

[9] G. J. Woeginger. Exact algorithms for NP-hard problems: A survey. In *Proc. 5th International Workshop on Combinatorial Optimization – Eureka, You Shrink!*, volume 2570 of *LNCS*, pages 185–208. Springer, 2003.