Approximability and parameterized complexity of multicover by c-intervals

René van Bevern[a,*], Jiehua Chen[a], Falk Hüffner[a], Stefan Kratsch[b], Nimrod Talmon[a], Gerhard J. Woeginger[c]

[a]*Institut für Softwaretechnik und Theoretische Informatik, TU Berlin, Germany*
[b]*Department of Computer Science, University of Bonn, Germany*
[c]*Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands*

Abstract

A c-interval is the disjoint union of c intervals over $\mathbb{N}$. The c-Interval Multicover problem is the special case of Set Multicover where all sets available for covering are c-intervals. We strengthen known APX-hardness results for c-Interval Multicover, show W[1]-hardness when parameterized by the solution size, and present fixed-parameter algorithms for alternative parameterizations.

Keywords: NP-hard problems, APX-hardness, fixed-parameter tractability, W[1]-hardness, set cover

1. Introduction

For a positive integer n, we consider the ground set $\{1, \dots, n\}$ whose elements we call *positions*. A *c-interval* is the disjoint union of c intervals. We study the following problem:

c-Interval Multicover
Input: A multiset $\mathcal{C}$ of c-intervals over $\{1, \dots, n\}$ and a demand function $d\colon \{1, \dots, n\} \to \mathbb{N}$.
Output: A minimum-size *cover* $\mathcal{S} \subseteq \mathcal{C}$ such that each position $i \in \{1, \dots, n\}$ is contained in at least $d(i)$ c-intervals of $\mathcal{S}$.

The c-Interval Multicover problem is a special case of the well-known Set Multicover problem, in which the multiset $\mathcal{C}$ may consist of arbitrary sets and is not constrained to c-intervals. Special cases and generalizations of c-Interval Multicover have been considered in the literature [3, 4, 10]:

(1) Ding, Fu, and Zhu [3, 4] investigate the special case m-Fold c-Interval Cover, where $d(i) = m$ for all $i \in \{1, \dots, n\}$. They are particularly interested in the case of $c = 2$ and $m = 1$, which they call Paired-End Interval Cover.

(2) Hochbaum and Levin [10] consider Multi-Shift Scheduling, a generalization where each input c-interval can be used multiple times in the cover. This problem can be modeled as c-Interval Multicover by duplicating each input c-interval $\max_{i\in\{1,\dots,n\}} d(i)$ times. This transformation works in polynomial time if the input multiset $\mathcal{C}$ is represented as an ordinary set that stores the multiplicity of each element in binary.

1-Interval Multicover can be solved in polynomial time by a greedy algorithm [10]. 1-Fold 2-Interval Cover is NP-hard even if each input 2-interval contains at most three positions [3, 4]. However, if each 2-interval contains at most two positions, then the problem is polynomial-time solvable by techniques from matching theory [3, 4, 10]. 1-Fold 3-Interval Cover is APX-hard due to a simple reduction from Set Cover [3, 4, 10].

In Section 2, we strengthen known results in the context of approximation algorithms [14]. In Section 3, we complement them by results in the context of parameterized complexity [6, 8, 13].

Related work. 1-Fold c-Interval Cover is a special case of Set Cover, in which the sets used for covering form c-intervals. More precisely, if we view a Set Cover instance as a binary matrix $A = (a_{i,j})$ with $a_{i,j} = 1$ if i is in the j-th set and $a_{i,j} = 0$ otherwise, then the goal is to select a minimum number of columns such that at least one 1 from each row is selected. In this setting, 1-Fold c-Interval Cover is the special case where each column contains at most c blocks of consecutive ones. The literature contains some hardness results and fixed-parameter algorithms for the special case where each *row* consists of blocks of consecutive ones [5, Section 4.2.2]. However, these results do not seem to transfer to our problem. Similarly, the fixed-parameter tractability results for problems on intersection graphs of c-intervals [7] do not seem to carry over to our problem, as an intersection graph cannot store the information on which c-intervals contain which positions.

2. Approximation

Ding et al. [4] give a factor-$2c(4m - 1)$ approximation algorithm for m-Fold c-Interval Cover. As noted earlier by Hochbaum and Levin [10], one can in fact get a factor-c approximation by splitting each c-interval into c intervals and solving the resulting m-Fold 1-Interval Cover instance optimally by using a greedy algorithm. The c-approximation of Hochbaum and Levin [10] even works for the more general c-Interval Multicover.

[*]Correspondence to: Technische Universität Berlin, Sekr. TEL 5-1, Ernst-Reuter-Platz 7, 10587 Berlin, Germany. Phone: +49 30 314 25312.
Email addresses: rene.vanbevern@tu-berlin.de (René van Bevern), jiehua.chen@tu-berlin.de (Jiehua Chen), falk.hueffner@tu-berlin.de (Falk Hüffner), kratsch@cs.uni-bonn.de (Stefan Kratsch), nimrodtalmon77@gmail.com (Nimrod Talmon), gwoegi@win.tue.nl (Gerhard J. Woeginger)

We note that it is possible to construct examples where the factor c is attained; thus, to improve the factor, it is necessary to improve the algorithm and not just its analysis.

2.1. A stronger APX-hardness result

1-Fold c-Interval Cover is APX-hard for $c \geq 3$ [10, 3, 4]. We strengthen this negative result to 1-Fold 2-Interval Cover. Our hardness result holds for very restricted types of 2-intervals, thus establishing a sharp separation line between easy and hard cases (recall that the case where each 2-interval consists of two length-1 intervals is polynomial-time solvable).

Theorem 1. 1-Fold 2-Interval Cover *is APX-hard even if each input c-interval is either*

- *a* 1*-interval of length* 3*, or*
- *a* 2*-interval consisting of length-1 intervals.*

Proof. We reduce from the following APX-hard problem [11]:

Maximum Bounded 3-Dimensional Matching
Input: Pairwise disjoint sets X, Y, Z and triples $T \subseteq X \times Y \times Z$ where every element of $X \cup Y \cup Z$ occurs in at most three triples of T.
Output: Find a maximum-size *matching* $T' \subseteq T$ such that no two triples in T' agree in any coordinate.

Given an instance $P = (X, Y, Z, T)$ of Maximum Bounded 3-Dimensional Matching, we construct an instance P' of 1-Fold 2-Interval Cover on a ground set of $|X \cup Y \cup Z| + 3|T|$ positions. First, for each $e \in X \cup Y \cup Z$, we allocate a position $A(e)$, which we call *type* A, and for each triple $t = (x, y, z) \in T$, we allocate three consecutive positions $B(t, x), B(t, y), B(t, z)$, which we call *type* B. The actual positions allocated are arbitrary except that the three positions for a triple are consecutive. Then, for each $e \in X \cup Y \cup Z$ and each $t \in T$, we add a 2-interval $I_{e,t}$ consisting of the two singleton intervals $A(e)$ and $B(t, e)$. Finally, for each $t = (x, y, z) \in T$, we add a 1-interval $I_t = \{B(t, x), B(t, y), B(t, z)\}$.

Claim 1. *Let $n := |X \cup Y \cup Z|$. Then, there is a matching $T' \subseteq T$ for P with $|T'| = \tau$ if and only if there is a cover $\mathcal{S}$ for P' with $|\mathcal{S}| = n + |T| - \tau$.*

Proof (of claim). ($\Rightarrow$) From a matching T' for P with $|T'| = \tau$, we can construct a cover $\mathcal{S}$ for P' as follows: first, for each $t \in T \setminus T'$, we choose the 1-interval I_t, and for each $t = (x, y, z) \in T'$, we choose the three 2-intervals $I_{x,t}, I_{y,t}, I_{z,t}$. This covers all positions of type B, and since the elements of T' are distinct in each coordinate, it also covers 3τ positions of type A. We then cover each of the $n - 3\tau$ still uncovered positions of type A with arbitrary 2-intervals. The cover has size $(|T| - \tau) + 3\tau + (n - 3\tau) = n + |T| - \tau$.

($\Leftarrow$) Consider any cover for P' of size $n + |T| - \tau$. We first canonicalize the cover without making it larger: while there is a position $A(e)$ for some $e \in X \cup Y \cup Z$ that is covered by a 2-interval $I_{e,t} = \{A(e), B(t, e)\}$ and another 2-interval, replace $I_{e,t}$ by I_t. We obtain a cover for P' of size at most $n + |T| - \tau$ where each position $A(e)$ is covered by exactly one 2-interval $I_{e,t}$. Thus, the cover contains at most $|T| - \tau$ of the intervals I_t. Consequently, at least τ of the intervals I_t are not contained in the cover. We now choose as a matching T' for P those τ triples t for which I_t is not in the cover. This is indeed a matching: if two selected triples t, t' were to coincide in an element e, then the positions $B(t, e)$ and $B(t', e)$ would need to be covered by two 2-intervals $I_{e,t}$ and $I_{e,t'}$, which is not possible after our preprocesing. $\diamond$

Now assume that we have an algorithm that approximates 1-Fold 2-Interval Cover within a factor of $(1 + \epsilon)$. If the optimal solution for P is τ^* and $n := |X \cup Y \cup Z|$, then we can find a solution of size τ for P with

$$n + |T| - \tau \leq (1 + \epsilon) \cdot (n + |T| - \tau^*). \qquad (*)$$

Since each element occurs in at most three triples, we have $|T| \leq 3n$. Moreover, each triple conflicts with at most six other triples in T, and hence $\tau^* \geq |T|/7$. Finally, after deleting elements from $X \cup Y \cup Z$ that are not contained in any triple, we have $3|T| \geq n$. This yields

$$\begin{aligned}(1 + \epsilon) \cdot \tau^* &\overset{(*)}{\leq} \tau + \epsilon \cdot (n + |T|) \leq \tau + 4\epsilon \cdot n \\ &\leq \tau + 4\epsilon \cdot 3|T| \leq \tau + 84\epsilon \cdot \tau^* \\ \implies \tau &\geq (1 - 83\epsilon) \cdot \tau^*.\end{aligned}$$

Thus, if 1-Fold 2-Interval Cover is approximable to an arbitrary factor, then also Maximum Bounded 3-Dimensional Matching is, implying APX-hardness for 1-Fold 2-Interval Cover. $\square$

3. Parameterized Complexity

Ding et al. [3, 4] note that if each position occurs in at most t c-intervals and $k := |\mathcal{S}|$ is the size of the solution, then m-Fold c-Interval Cover can be solved in $t^k \cdot n^{O(1)}$ time by a branching algorithm. That is, it is fixed-parameter tractable with respect to the combined parameter (k, t). Regarding tractability for parameter k alone, we prove that already 1-Fold 2-Interval Cover is W[1]-hard when parameterized by k (Section 3.1), thus not fixed-parameter tractable unless FPT = W[1]; only for very restricted instances we obtain tractability (Section 3.2). We obtain another positive result for a cutwidth-like parameter (Section 3.3).

3.1. Parameter "solution size"

Theorem 2. 1-Fold 2-Interval Cover *is W[1]-hard with respect to the parameter "number k of c-intervals to be selected".*

Proof. We prove W[1]-hardness by a parameterized reduction from the W[1]-hard Multicolored Clique problem [7] to 1-Fold 2-Interval Cover.

Multicolored Clique
Input: A graph $G = (V, E)$, an integer $k \in \mathbb{N}$, and vertex colors $\phi \colon V \to [1, k]$.
Question: Is there a vertex subset $V' \subseteq V$ that induces a clique in G and contains exactly one vertex of each color?

Let (G, ϕ, k) be an instance of Multicolored Clique, where $G = (V, E)$. For convenience, let $V = \{1, \ldots, n\}$ and $E = \{e_1, \ldots, e_m\}$. We construct an instance $(\mathcal{C}, k')$ of 1-Fold 2-Interval Cover that is yes if and only if (G, ϕ, k) is yes for Multicolored Clique.

That is, the instance $(\mathcal{C}, k')$ will allow for a cover with at most k' 2-intervals if and only if G contains a clique C of size at least k and $\phi(C) = [1, k]$.

We will construct two types of gadgets: *Edge selection gadgets* and *testing gadgets*, following the established strategy of modeling Multicolored Clique as the task of selecting one edge for each color pair and then checking *consistency*, that is, whether the selected edges select only one vertex of each color. Each gadget will have its private set of positions, but, for convenience, in the construction of each gadget we always start with position 0. The understanding is that we can turn all these position sets into a single ground set of consecutive positions $[0, N]$ by using appropriate offsets. All constructed intervals will be fully contained in the position set of a single gadget but 2-intervals may be composed of intervals from different gadgets. Finally, we will shorthand $K := 4(k - 2)$, which will turn out to be the number of copies that we have to make for each edge selection in order to test for consistency.

Edge selection gadgets. For each color combination $\{i, j\} \subseteq [1, k]$ we create an edge selection gadget $S(\{i, j\})$ as follows. The set of positions is $[0, m \cdot K + 1]$. For each $e_\ell = \{u, v\} \in E$ with $\{\phi(u), \phi(v)\} = \{i, j\}$, that is, for every edge with endpoints of color i and j, we introduce a 2-interval

$$I_{\{i,j\},\ell} = [0, (\ell - 1) \cdot K] \uplus [\ell \cdot K + 1, m \cdot K + 1].$$

In other words, the 2-interval $I_{\{i,j\},\ell}$ contains all positions of the gadget except for K consecutive positions in $[(\ell-1) \cdot K+1, \ell \cdot K]$. Moreover, these K positions are disjoint for every two different of these 2-intervals. We will use the requirement to cover these positions to test the consistency of the $\binom{k}{2}$ edge selections. For the moment, let us only introduce singleton intervals $[s, s]$ for all $s \in [1, m \cdot K]$ to use them later for 2-intervals; note that there are no such singletons for the first and last position in the gadget.

Testing gadgets. The following testing gadgets test the consistency of the edge selections. Concretely, if we choose edge e_ℓ for colors i and j and edge $e_{\ell'}$ for colors i and j', then we want the endpoints of color i in both edges to be the same. For technical reasons we make *two gadgets* $T(j, i, j')$ and $T(j', i, j)$ for each such test, that is, there will be $k(k - 1)(k - 2)$ testing gadgets.

We explain how to construct $T(j, i, j')$ for $\{i, j, j'\} \subseteq [1, k]$: The set of positions is $[0, n + 1]$. For every edge $e_\ell = \{u, v\}$ with $\phi(u) = j$ and $\phi(v) = i$ we introduce an interval $[0, v]$ (recall that $v \in V = [1, n]$). For every edge $e_{\ell'} = \{v', w\}$ with $\phi(v') = i$ and $\phi(w) = j'$ we introduce an interval $[v' + 1, n + 1]$. It is important to note the asymmetry of this construction, that is, we are checking the interaction of edges from colors j and j' to color i but the ordering of j and j' in $T(j, i, j')$ is relevant. The idea is that the intervals $[0, v]$ and $[v' + 1, n + 1]$ together cover all positions if and only if $v' \leq v$. Moreover, since we also create the gadget $T(j', i, j)$, where the roles will be exchanged, we can cover both sets of positions only with intervals belonging to e_ℓ and $e_{\ell'}$ if and only if $v = v'$. Otherwise, if $v \neq v'$ then using only these intervals will give an overlap of the intervals in one of $T(j, i, j')$ and $T(j', i, j)$, and an uncovered gap in the other.

Connecting the gadgets. At this point we have all *intervals* that we need for our construction but we still need to combine them into 2-*intervals*. Concretely, let $e_\ell = \{u, v\}$ be any edge and assume $\{\phi(u), \phi(v)\} = \{i, j\}$. We have intervals associated with e_ℓ in the testing gadgets $T(j, i, j')$ and $T(j', i, j)$ for $j' \in [1, k] \setminus \{i, j\}$, that is, for checking consistency for color i, and in testing gadgets $T(i, j, i')$ and $T(i', j, i)$ for $i' \in [1, k] \setminus \{i, j\}$, that is, for checking consistency for color j. Overall, these are $4(k - 2) = K$ intervals, matching the K singleton intervals that are left uncovered when we pick interval $I_{\{i,j\},\ell}$ corresponding to e_ℓ in selection gadget $S(\{i, j\})$. We recall that these positions are disjoint for different choices of edges in $S(\{i, j\})$. Thus, we may arbitrarily pair up the singleton intervals with intervals in the mentioned K testing gadgets. Together with the 2-intervals previously created in the selection gadgets (and modulo arranging all gadgets such that we get a consecutive sequence of positions) this defines the final set $\mathcal{C}$ of 2-intervals.

Setting the budget. The budget k' is set to $\binom{k}{2} + 2k(k - 1)(k - 2)$. This is intended to be used by picking exactly one interval $I_{\{i,j\},\ell}$ in each of the $\binom{k}{2}$ selection gadgets $S(\{i, j\})$ and picking exactly two intervals in each testing gadget $T(j, i, j')$. It is easy to see that this is correctly enforced: Singleton intervals in $S(\{i, j\})$ do not contain positions 0 or $m \cdot K + 1$, forcing selection of at least one interval $I_{\{i,j\},\ell}$. Similarly, in any testing gadget $T(j, i, j')$ we have no interval containing both position 0 and position $n + 1$ and, thus, require at least two intervals to be chosen. Hence, we already need at least k' 2-intervals and may not select any further ones. Whether or not the created instance has a cover with k' 2-intervals thus depends on the singleton intervals that are paired up with intervals in testing gadgets.

This completes the construction. It can be checked that it can be implemented to work in polynomial time. The output is the instance $(\mathcal{C}, k')$ of 1-Fold 2-Interval Cover.

Correctness. ($\Rightarrow$) Assume first that (G, ϕ, k) is yes for Multicolored Clique and let $C = \{v_1, \ldots, v_k\} \subseteq V$ be a clique in G with $\phi(v_i) = i$. We will construct a size-k' cover for the created 1-Fold 2-Interval Cover instance. In the selection gadgets $S(\{i, j\})$, we pick the 2-interval $I_{\{i,j\},\ell(\{i,j\})}$ with $e_{\ell(\{i,j\})} = \{v_i, v_j\}$. Additionally, we pick all 2-intervals that contain the K singletons that are not covered by $I_{\{i,j\},\ell(\{i,j\})}$ in $S(\{i, j\})$. This yields exactly

$$\binom{k}{2} \cdot (1 + K) = \binom{k}{2} + \binom{k}{2} \cdot (4(k-2)) = \binom{k}{2} + 2k(k-1)(k-2) = k'$$

2-intervals $\mathcal{S} \subseteq \mathcal{C}$, which clearly covers all positions in edge selection gadgets $S(\{i, j\})$.

Now consider any testing gadget $T(j, i, j')$ for $\{i, j, j'\} \subseteq [1, k]$. We selected all 2-intervals containing singletons for edges $\{v_i, v_j\}$ and $\{v_i, v_{j'}\}$ in the respective selection gadgets $S(\{i, j\})$ and $S(\{i, j'\})$. Thus, by construction, these 2-intervals also contain the intervals associated with these edges in $T(j, i, j')$. Concretely, these intervals are $[0, v_i]$ and $[v_i + 1, n + 1]$ (noting that in the construction we have $u = v_j$, $v = v_i$, $v' = v_i$, and $w = v_{j'}$). Thus, all positions in $T(j, i, j')$ are covered. It follows that $\mathcal{S}$ is indeed a solution for the instance $(\mathcal{C}, k')$, as claimed.

($\Leftarrow$) Now assume that we have a size-k' cover $\mathcal{S} \subseteq \mathcal{C}$ for the created 1-Fold 2-Interval Cover instance. We already know that it contains exactly one 2-interval $I_{\{i,j\},\ell(\{i,j\})}$ of each selection gadget $S(\{i,j\})$, so let $e_{\ell(\{i,j\})}$ be the corresponding edges of G. By construction, these edges have the correct endpoint colors since that was prerequisite for adding 2-intervals to $S(\{i,j\})$. It also follows that $\mathcal{S}$ contains all 2-intervals that contain the K singleton intervals left uncovered by $I_{\{i,j\},\ell(\{i,j\})}$ in $S(\{i,j\})$ since only one 2-interval $I_{\{i,j\},\cdot}$ can be afforded in budget k'. In total, we find that the choices of $I_{\{i,j\},\ell(\{i,j\})}$ and the implied singletons already account for the total budget: There are $\binom{k}{2}$ gadgets $S(\{i,j\})$ and in each we pick a 2-interval and K singletons, which gives a total of k' 2-intervals (by the same computation as above). We are now prepared to prove the consistency of the edge selections.

Pick any color combination $\{i,j,j'\} \subseteq [1,k]$, let $e_{\ell(\{i,j\})} = \{v,u\}$ and $e_{\ell(\{i,j'\})} = \{v',w\}$ with $\phi(v) = \phi(v') = i$, $\phi(u) = j$ and $\phi(w) = j'$. We want to establish $v = v'$. Assume that this is not the case. Then, w.l.o.g., we have $v < v'$. Consider the testing gadget $T(j,i,j')$, which, by assumption, is covered by $\mathcal{S}$. By tightness of the budget, we know that $\mathcal{S}$ contains exactly two intervals in $T(j,i,j')$. Since $v < v'$, the cover $\mathcal{S}$ cannot use exactly the two intervals corresponding to $e_{\ell(\{i,j\})}$ and $e_{\ell(\{i,j'\})}$: By construction, the corresponding intervals are $[0,v]$ and $[v'+1,n+1]$ and fail to cover position v'. This, however, yields a contradiction since $\mathcal{S}$ can only use such intervals in $T(j,i,j')$ that are paired up with singletons corresponding to $e_{\ell(\{i,j\})}$ or $e_{\ell(\{i,j'\})}$. It follows that $v = v'$.

Thus, the $\binom{k}{2}$ edges $e_{\ell(\{i,j\})}$ agree on all their endpoints of the same color. Hence, there is only one endpoint of each color and the edges are the edge set of a k-clique in G containing vertices of all k colors (as given by ϕ). Thus, (G,ϕ,k) is yes, as claimed.

Let us wrap up the proof. We gave a reduction from Multicolored Clique to 1-Fold 2-Interval Cover that can be implemented in polynomial time and produces an equivalent instance with a parameter value k' that depends only on the input parameter k. This fulfills the requirements of a parameterized reduction and implies W[1]-hardness of 1-Fold 2-Interval Cover. □

3.2. *1-intervals and short c-intervals*

In contrast to Theorem 2, we can show that 1-Fold 2-Interval Cover is fixed-parameter tractable when combining the parameter k with the parameter "maximum number C of positions contained in any non-1-interval".

Theorem 3. *Given an upper bound k on the size of an optimal solution and assuming each input c-interval to be either*

- *a 1-interval, or*
- *to contain at most C positions,*

2-Interval Multicover *is solvable in $O(k^k \cdot (kC)^{kC} \cdot |\mathcal{C}|)$ time.*

Our algorithm for Theorem 3 makes use of two simple data reduction rules. The first data reduction rule exploits the fact that a solution of size at most k can contain at most k copies of any input c-interval.

Reduction Rule 1. If a c-interval occurs in the input more than k times, then delete one of its copies.

The second data reduction rule exploits the fact that, if some position i has a demand $d(i) \leq 0$, then the size of an optimum solution does not change if we delete this position from the input.

Reduction Rule 2. If there is a position i with $d(i) \leq 0$, then compute new demands $d' \colon \{1,\dots,n-1\} \to \mathbb{Z}$,

$$d'(j) \mapsto \begin{cases} d(j) & \text{if } j < i \\ d(j+1) & \text{if } j \geq i, \end{cases}$$

replace each input c-interval $I \in \mathcal{C}$ by the new c-interval

$$I' := \{j < i \mid j \in I\} \cup \{j \geq i \mid j+1 \in I\},$$

and decrement n by one.

Finally, our algorithm for Theorem 3 exploits the following observation.

Observation 1. If a solution $\mathcal{S}$ contains a 1-interval $I = [j,\ell]$ and there is a 1-interval $I' = [j,\ell'] \notin \mathcal{S}$ with $\ell' > \ell$, then there is a solution with the same size as $\mathcal{S}$ and containing I' instead of I.

Proof (of Theorem 3). Given a natural number k, the algorithm tries to find a solution of size at most k using a simple branching strategy. In each step of the algorithm, we assume the input to be completely reduced with respect to Reduction Rules 1 and 2. This can always be achieved in $O(n \cdot |\mathcal{C}|)$ time. Now, start with position $i := 1$ and execute the following steps.

(1) Branch into all possibilities for choosing the first starting position $i' \geq i$ of a 1-interval in an optimal solution. Since all positions have positive demand, the positions before i' have to be covered by non-1-intervals. Thus, $i' \leq i + kC$, since a non-1-interval contains at most C positions and an optimal solution contains at most k c-intervals. It follows that we branch into at most kC possibilities for i'.

(2) Branch into all possibilities for choosing the number $1 \leq z \leq k$ of 1-intervals starting at position i' in an optimal solution.

(3) Compute the set S of the z longest intervals starting in i'. By Observation 1, we can assume these to be part of an optimal solution. Hence, add the intervals in S to our solution, delete them from our input instance, decrement k by z and decrement each demand $d(i)$ by the number of intervals in S containing i.

(4) If $i < n$ and $k > 0$, then increment i by one and go to (1).

(5) If $i \geq n$ and $k > 0$, then branch into all possibilities of choosing a non-1-interval I into the solution. In each case, decrement k and the demands of the positions contained in I by one and repeat (5).

The algorithm is correct since, for each position i, we try all possibilities for the number $z \leq k$ of 1-intervals starting at i in an optimal solution and, by Observation 1, make an optimal choice among these z 1-intervals. When all 1-intervals are fixed,

(5) tries all possibilities of covering the remaining positions using at most k non-1-intervals.

It remains to analyze the running time of the algorithm. Clearly, each search tree node can be processed in $O(n \cdot |\mathcal{C}|)$ time. For the number of search tree nodes, notice that step (1) followed by (2) branches into at most k^2C cases, in each of which k is decremented by at least one.

In (5) all remaining positions with positive demand have to be covered by at most k non-1-intervals, each containing at most C positions. Hence, there are $n \leq kC$ positions left or the algorithm can answer "no" in this branch. Since each remaining non-1-interval occurs at most k times, there are at most $k \cdot (kC)^C$ non-1-intervals left, which is the number of cases that step (5) branches into.

Since the depth of the search tree is bounded by k, it has $O(k^k \cdot (kC)^{kC})$ nodes, each of which can be processed in $O(n \cdot |\mathcal{C}|)$ time. □

3.3. Cutwidth-like parameters

We now consider the parameter "maximum number Q of c-intervals live at any position".

Definition 1. A c-interval I is *live at position* $i \in \mathbb{N}$ if there are $j, k \in I$ with $j \leq i \leq k$.

Note that a c-interval does not have to contain position i in order to be live at position i. Similar parameters of "live objects" have been previously exploited in interval scheduling tasks [1, 9].

We show that c-Interval Multicover is solvable in $O(4^Q Q \cdot n)$ time if each position has at most Q live c-intervals. Our algorithm actually solves the more general Set Multicover:

Set Multicover (SMC)
Input: A universe U, a multiset $\mathcal{C} \subseteq 2^U$, and a demand function $d\colon U \to \mathbb{N}$.
Output: Find a minimum-size *set multicover* $\mathcal{S} \subseteq \mathcal{C}$ that covers each element $u \in U$ at least $d(u)$ times.

We show that Set Multicover can be solved in $O(4^t t \cdot n)$ time if a linear layout of cutwidth at most t of its universe is given.

Definition 2. A *linear layout of cutwidth at most* t for a Set Multicover instance $(U, \mathcal{C})$ is an injective function $\ell : U \to \mathbb{N}$ such that, for each $i \in \mathbb{N}$, the multiset $\mathcal{C}_i := \{C \in \mathcal{C} \mid \exists u, w \in C : \ell(u) \leq i \leq \ell(w)\}$ contains at most t elements.

It can be decided in linear time whether a Set Multicover instance allows for a linear layout of constant cutwidth t [2]. Although it is open whether the corresponding layout is constructible in the same time, we exploit the fact that a c-Interval Multicover instance with at most Q live c-intervals immediately yields a Set Multicover instance with a linear layout of cutwidth at most Q. Thus, the following theorem becomes applicable.

Theorem 4. Set Multicover *is solvable in* $O(4^t t \cdot n)$ *time if a linear layout of cutwidth at most t is given.*

Proof. We use dynamic programming. Given an Set Multicover instance $(U, \mathcal{C})$, we want to compute a minimum cover $\mathcal{S} \subseteq \mathcal{C}$. Without loss of generality, we assume $U := \{1, 2, \dots, n\}$ to be labeled in the order of the given linear layout. By assumption, for all $i \in \mathbb{N}$, we have $|\mathcal{C}_i| \leq t$.

For all $\mathcal{S}' \subseteq \mathcal{C}_i$ and $i \in \{1, \dots, n\}$, we define $T[i, \mathcal{S}']$ to be the size of a minimum-size $\mathcal{S} \subseteq \mathcal{C}$ covering each element $j \in \{1, \dots, i\}$ at least $d(j)$ times and satisfying $\mathcal{S} \cap \mathcal{C}_i = \mathcal{S}'$. Obviously $T[1, \mathcal{S}'] = |\mathcal{S}'|$ for all $\mathcal{S}' \subseteq \mathcal{C}_1$ that cover position 1 at least $d(1)$ times. For all other $\mathcal{S}' \subseteq \mathcal{C}_1$, we set $T[1, \mathcal{S}'] = \infty$. Now, for all $i \in \{1, \dots, n-1\}$ and $\mathcal{S}' \subseteq \mathcal{C}_{i+1}$, we show by induction that

$$T[i+1, \mathcal{S}'] = \begin{cases} \infty \text{ if } \mathcal{S}' \text{ covers } i+1 \text{ less than } d(i+1) \text{ times,} \\ \min\limits_{\substack{\mathcal{S}'' \subseteq \mathcal{C}_i \\ \mathcal{S}'' \cap \mathcal{C}_{i+1} = \mathcal{S}' \cap \mathcal{C}_i}} (T[i, \mathcal{S}''] + |\mathcal{S}' \setminus \mathcal{C}_i|) \quad \text{otherwise.} \end{cases}$$

We first show that any set $\mathcal{S}$ that covers $\{1, \dots, i+1\}$ satisfies $|\mathcal{S}| \geq T[i+1, \mathcal{S}']$, where $\mathcal{S}' = \mathcal{S} \cap \mathcal{C}_{i+1}$. Observe that $Z := (\mathcal{S} \setminus \mathcal{C}_{i+1}) \cup (\mathcal{S} \cap \mathcal{C}_i)$ covers $\{1, \dots, i\}$. By the induction hypothesis, for $\mathcal{S}'' := Z \cap \mathcal{C}_i$, we have $|Z| \geq T[i, \mathcal{S}'']$. Moreover, $\mathcal{S}'' \cap \mathcal{C}_{i+1} = \mathcal{S} \cap \mathcal{C}_i \cap \mathcal{C}_{i+1} = \mathcal{S}' \cap \mathcal{C}_i$. Thus, by definition, $T[i+1, \mathcal{S}'] \leq T[i, \mathcal{S}''] + |\mathcal{S}' \setminus \mathcal{C}_i| \leq |Z| + |\mathcal{S}' \setminus \mathcal{C}_i| = |Z \cup (\mathcal{S}' \setminus \mathcal{C}_i)| = |\mathcal{S}|$.

We now show that there is a set $\mathcal{S}$ with $\mathcal{C}_{i+1} \cap \mathcal{S} = \mathcal{S}'$ that covers $\{1, \dots, i+1\}$ and contains at most $T[i+1, \mathcal{S}']$ elements. If $T[i+1, \mathcal{S}'] \neq \infty$, then $T[i+1, \mathcal{S}'] = T[i, \mathcal{S}''] + |\mathcal{S}' \setminus \mathcal{C}_i|$ for some $S'' \subseteq \mathcal{C}_i$. Thus, by the induction hypothesis, there is a set Z that covers $\{1, \dots, i\}$ and contains at most $T[i, \mathcal{S}'']$ elements. Moreover, $\mathcal{S}'$ covers position $i+1$ at least $d(i+1)$ times. Thus, $Z \cup \mathcal{S}'$ covers $\{1, \dots, i+1\}$. We next compute its size. Without loss of generality, we assume that $Z \cap (\mathcal{C}_{i+1} \setminus \mathcal{C}_i) = \emptyset$ because sets in $\mathcal{C}_{i+1} \setminus \mathcal{C}_i$ cannot cover positions in $\{1, \dots, i\}$. Hence, $|Z \cup \mathcal{S}'| = |Z \uplus (\mathcal{S}' \setminus \mathcal{C}_i)| \leq T[i, \mathcal{S}''] + |\mathcal{S}' \setminus \mathcal{C}_i|$.

Using the above recurrence, we can compute $T[n, \mathcal{S}']$ for all $\mathcal{S}' \subseteq \mathcal{C}_n$ in $O(4^t t \cdot n)$ time. The optimal solution to the Set Multicover instance is $\arg\min_{\mathcal{S}' \subseteq \mathcal{C}_n} T[n, \mathcal{S}']$. □

Since a c-Interval Multicover instance with at most Q live c-intervals is a Set Multicover instance with a linear layout of cutwidth at most Q, we obtain the following corollary.

Corollary 1. *If there are at most Q c-intervals live at any position, then c-*Interval Multicover *is solvable in $O(4^Q Q \cdot n)$ time.*

4. Conclusion

We studied special cases of the well-known Set Multicover problem. Since we showed that even very restricted variants of the problem remain APX-hard and since a simple factor-c approximation for c-Interval Multicover is known [10], a canonical question is whether this factor c can be improved. Herein, fixed-parameter approximation algorithms could be useful [12].

For parameterized algorithms, an interesting parameter to investigate in future research would be the maximum distance between the first and last position contained in a c-interval.

Finally, it would be interesting to consider the related exact cover and packing variants of c-Interval Multicover, that is, finding a subset of c-intervals covering each position i *exactly* $d(i)$ times or finding a maximum-size subset covering each position i *at most* $d(i)$ times.

Acknowledgments

René van Bevern was supported by the Deutsche Forschungsgemeinschaft (DFG) project DAPA (NI 369/12), Falk Hüffner by DFG project ALEPH (HU 2139/1), Nimrod Talmon by DFG Research Training Group "Methods for Discrete Structures" (GRK 1408), and Gerhard Woeginger by the Alexander von Humboldt Foundation, Bonn, Germany. The main work of Stefan Kratsch was done while affiliated with TU Berlin and supported by DFG project PREMOD (KR 4286/1).

[1] R. van Bevern, M. Mnich, R. Niedermeier, and M. Weller. Interval scheduling and colorful independent sets. *J. Sched.*, 2014. In press.

[2] R. van Bevern, R. G. Downey, M. R. Fellows, S. Gaspers, and F. A. Rosamond. Myhill-Nerode methods for hypergraphs. *Algorithmica*, 2015. Accepted for publication, preprint available via arXiv:1211.1299v5.

[3] L. Ding, B. Fu, and B. Zhu. Minimum interval cover and its application to genome sequencing. In *Proc. 5th COCOA*, volume 6831 of *LNCS*, pages 287–298. Springer, 2011.

[4] L. Ding, B. Fu, and B. Zhu. Minimum paired-end interval cover and its application. Manuscript, May 2012. URL http://www.cs.montana.edu/bhz/doc/tcbb12-1.pdf.

[5] M. Dom. Algorithmic aspects of the consecutive-ones property. *Bulletin of the EATCS*, 98:27–59, 2009.

[6] R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013.

[7] M. R. Fellows, D. Hermelin, F. Rosamond, and S. Vialette. On the parameterized complexity of multiple-interval graph problems. *Theor. Comput. Sci.*, 410(1):53–61, 2009.

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

[9] M. M. Halldórsson and R. K. Karlsson. Strip graphs: Recognition and scheduling. In *Proc. 32nd WG*, volume 4271 of *LNCS*, pages 137–146, 2006.

[10] D. S. Hochbaum and A. Levin. Cyclical scheduling and multi-shift scheduling: Complexity and approximation algorithms. *Discrete Optim.*, 3(4): 327–340, 2006.

[11] V. Kann. Maximum bounded 3-dimensional matching is MAX SNP-complete. *Inform. Process. Lett.*, 37(1):27–35, 1991.

[12] D. Marx. Parameterized complexity and approximation algorithms. *Comput. J.*, 51(1):60–78, 2008.

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

[14] D. P. Williamson and D. B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.