JOURNAL OF COMPUTATIONAL BIOLOGY
Volume 25, Number 11, 2018
Mary Ann Liebert, Inc.
Pp. 1171–1178
DOI: 10.1089/cmb.2018.0032

Research Articles

Joker de Bruijn: Covering *k*-Mers Using Joker Characters

YARON ORENSTEIN,[1,2] YUN WILLIAM YU,[3] and BONNIE BERGER[2,3]

ABSTRACT

Sequence libraries that cover all *k*-mers enable universal and unbiased measurements of nucleotide and peptide binding. The shortest sequence to cover all *k*-mers is a de Bruijn sequence of length $|\Sigma|^k + k - 1$. Researchers would like to increase *k* to measure interactions at greater detail, but face a challenging problem: the number of *k*-mers grows exponentially in *k*, while the space on the experimental device is limited. In this study, we introduce a novel advance to shrink *k*-mer library sizes by using joker characters, which represent all characters in the alphabet. Theoretically, the use of joker characters can reduce the library size tremendously, but it should be limited as the introduced degeneracy lowers the statistical robustness of measurements. In this work, we consider the problem of generating a minimum-length sequence that covers a given set of *k*-mers using joker characters. The number and positions of the joker characters are provided as input. We first prove that the problem is NP-hard. We then present the first solution to the problem, which is based on two algorithmic innovations: (1) a greedy heuristic and (2) an integer linear programming (ILP) formulation. We first run the heuristic to find a good feasible solution, and then run an ILP solver to improve it. We ran our algorithm on DNA and amino acid alphabets to cover all *k*-mers for different values of *k* and *k*-mer multiplicity. Results demonstrate that it produces sequences that are very close to the theoretical lower bound.

Keywords: de Bruijn sequence, microarray library design, peptide arrays, protein binding, protein binding microarrays.

1. INTRODUCTION

Protein-DNA, -RNA, and -peptide interactions drive nearly all cellular processes. Protein-DNA binding regulates gene expression by binding to specific DNA sequences; protein-RNA interactions regulate gene expression post-transcriptionally by stabilizing, splicing, and degrading RNA; and protein-peptide interactions are key for cellular signaling in vivo.

[1]Department of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Beer-Sheva, Israel.
[2]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, Massachusetts.
[3]Department of Mathematics, Massachusetts Institute of Technology, Cambridge, Massachusetts.

© Yaron Orenstein, et al., 2018. Published by Mary Ann Liebert, Inc. This Open Access article is distributed under the terms of the Creative Commons Attribution Noncommercial License (http://creativecommons.org/licenses/by-nc/4.0/) which permits any noncommercial use, distribution, and reproduction in any medium, provided the original author(s) and the source are credited.

High-throughput experimental data describing the strength and specificity for individual proteins interacting with universal unbiased libraries provide critical information required to reconstruct interaction networks. Such a measurement can be achieved by directly measuring binding to sequence libraries that cover a large space of DNA, RNA, or amino acid k-mers. The comprehensive coverage guarantees that specificities can be identified de novo for any protein. Microarrays that cover all k-mers have been used successfully in various technologies to measure protein-DNA, -RNA, and -peptide binding. In Table 1, we summarize the specifications of five such technologies (Berger et al., 2006; Fordyce et al., 2010; Gurard-Levin et al., 2010; Ray et al., 2013; Smith et al., 2013).

While these technologies have been used successfully to measure protein interactions, they all face a similar challenge: space on the experimental device and the sequence length that can be used are both limited, restricting the total sequence space that can be probed in a single experiment. In particular, increasing k poses difficulties since the number of sequences needed to cover all k-mers increases exponentially with k as the number of k-mers over alphabet Σ is $|\Sigma|^k$.

Several solutions have been suggested to generate sequence libraries that cover all possible k-mers in the most compact space possible. A de Bruijn sequence is the shortest sequence, in which each k-mer appears exactly once. Its length is given by $|\Sigma|^k + k - 1$. De Bruijn sequences were used in protein-binding microarrays for $k = 10$ (Philippakis et al., 2008). A reduction of DNA libraries by half was achieved by utilizing the reverse complementarity property of double-stranded DNA (D'Addario et al., 2012; Orenstein and Shamir, 2013; Smith et al., 2013). Other methods produce compact, unstructured RNA libraries to measure protein-RNA binding (Ray et al., 2013; Orenstein and Berger, 2015). However, in all solutions, all k-mers have to occur in the sequence set, thus limited by the number of k-mers $|\Sigma|^k$.

In this study, we introduce a novel idea to generate smaller libraries to cover a given set of k-mers by using *joker* characters. Joker characters represent degenerate nucleotides (or amino acids) covering all characters in the alphabet, that is, joker character x representing $\{A, C, G, T\}$. Such degenerate nucleotides (or amino acids) can be ordered directly from the vendor during oligonucleotide (or peptide) synthesis at no extra cost, providing a new potential avenue for probing a larger sequence space within the constraints of limited experimental space.

The downside of using joker characters is that they introduce degeneracy, which lowers the statistical robustness of measurements: a measurement of a single microarray spot is now assigned to multiple sequences instead of just one. In the extreme case, a sequence of k consecutive joker characters covers all k-mers, but produces only a single measurement, which is useless for inferring protein-binding specificities. To rectify this problem, we set a limit to the use of joker characters by having the user provide the number and positions of joker characters in the sequence.

Previous studies have considered the problem of covering k-mers using joker characters. Blanchet-Sadri et al. (2010) solved the problem of covering all binary k-mers with exactly one joker character. In the thesis by Wyatt (2013), a solution was given to the problem of covering all binary k-mers with multiple joker characters, but with no other restrictions. Last, Chen et al. (2016) studied the problem of covering all binary k-mers with a few joker characters, but required that each k-mer appears exactly once and with no other restrictions. None considered the coverage of a given set of k-mers with a limitation on the number and positions of joker characters.

In this work, we study the problem of generating a minimum-length sequence to cover a given set of k-mers with a given number and positions of joker characters. We first prove that the problem is NP-hard. We then describe a novel greedy heuristic, which finds a sequence in time polynomial in the output length. Then,

Table 1. Specifications of Technologies Designed to Cover All k-Mers by k-Mer Value, Alphabet, Probe Sequence Length, and Number of Sequences

Technology	*PBM*	*MITOMI*	*Synthetic enhancers*	*RNAcompete*	*Peptide arrays*
k	10	8	6	9	2
Alphabet	DNA	DNA	DNA	RNA	Amino acid
Sequence length	35	52	15	30–41	2
No. of sequences	40,330	1457	184	241,357	361

MITOMI, mechanically-induced trapping of molecular interaction; PBM, protein binding microarrays.

we formulate the problem as an integer linear programming (ILP) problem to produce an optimal solution. We suggest a two-step approach: running the greedy heuristic and improving its solution using an ILP solver. We compare our results with theoretical lower bounds and a random approach. The implementation of our algorithm is freely available at jokercake.csail.mit.edu

2. PRELIMINARIES

A *k-mer* is a word of length k over a given alphabet Σ. In this study, we refer to two alphabets $\Sigma_{AA} = \{A, R, N, D, C, Q, E, G, H, I, L, K, M, F, P, S, T, W, Y, V\}$ and $\Sigma_{DNA} = \{A, C, G, T\}$. We interchangeably refer to a k-mer as a word and an integer by the natural conversion in base $|\Sigma|$. A *joker* character, denoted by x, represents all characters in Σ, that is, x representing $\{A, C, G, T\}$. K-mer $w = (w_1, \ldots, w_k)$ is *covered* by sequence S if there exists $0 \leq i \leq |S| - k$ such that for $1 \leq j \leq k$: $S_{i+j} \in \{x, w_j\}$. We say that w *occurs* at index i in S. In other words, any original character of w may be replaced by the joker character.

We define two new notations relating to k-mer coverage with joker characters. *Template* t is a k-mer over $\{0,1\}$, where 1 denotes joker positions. Sequence S follows template t if its joker positions are the 1 positions in a concatenation of multiple templates t. Denote $|t|_1$ as the weight of template t, that is, the number of 1s in it. For example, $S = AxCCGxTA$ follows template $t = 0100$ and $|t|_1 = 1$. We denote $S \in_t [\Sigma \cup \{x\}]^\ell$, where in the example, $\ell = 2|t|$. *K-mer counts* C is a vector over natural values of length $|\Sigma|^k$. Element $C(w)$ corresponds to the number of times k-mer w is covered by the sequence. K-mer w is covered at least $C(w)$ times by sequence S if there are $p \geq C(w)$ distinct indices $\{i_1, \ldots, i_p\}$ such that w occurs at index i_j in S for $1 \leq j \leq p$. Using the above notations, we define a (C, t, Σ)-joker de Bruijn sequence as a sequence covering k-mers according to C following template t.

We also define *reverse complementarity*. A complement relation is a symmetric nonreflexive relation, that is, $\overline{A} = T$ and $\overline{C} = G$. The reverse complement of k-mer $w = \{w_1, \ldots, w_k\}$ is $RC(w) = \{\overline{w_k}, \ldots, \overline{w_1}\}$. A k-mer is *RC covered* by sequence S if it occurs in either S or $RC(S)$. A (C, t, Σ)-RC-joker de Bruijn sequence *RC* covers k-mers according to C and follows template t.

In this study, we consider the following problem and its version utilizing the reverse complement property.

MINIMUM-LENGTH (C, t, Σ)-JOKER DE BRUIJN SEQUENCE
INSTANCE: k-met counts C, template t, alphabet Σ.
VALID SOLUTION: (C, t, Σ)-joker de Bruijn sequence S.
GOAL: Minimize $|S|$.

3. METHODS

3.1. Greedy heuristic

We present a novel algorithm to find a (C, t, Σ)-joker de Bruijn sequence. It is based on a greedy heuristic that examines at each step an addition of k characters from $\Sigma \cup \{x\}$ that follow template t. The addition that covers the most k-mers that are yet to be covered (including multiple k-mer instances if needed) is chosen and added to the current sequence. The algorithm terminates when all k-mers have been covered according to C. The algorithm is summarized as Algorithm 1.

We bound the runtime of Algorithm 1. We first prove the following Lemma on the minimum number of k-mers covered in each iteration of the top while loop (line 4 in Algorithm 1).

Lemma 1. *In each iteration of the while loop in Algorithm 1, at least one k-mer has an increased k-mer count.*

Proof. Denote w as a k-mer for which $A(w) < C(w)$. The inner for loop (line 6) iterates over all possible k-mers that follow template t, including those that cover w. Denote w_t as k-mer w with jokers in 1 positions of t. It follows t and covers w. Thus, $CURR_{2:k} \cdot w_t$ adds one to the coverage of w. Since the for loop finds the maximum, it has to be at least one.

Corollary 1. *The number of iterations of the while loop in Algorithm 1 is bounded by* $\sum_{i=0}^{|\Sigma|^k-1} C(i)$.

Proof. The number of required k-mer coverages is $\sum_{i=0}^{|\Sigma|^k-1} C(i)$. By Lemma 1, at least one k-mer has an increased count at each iteration. Thus, the bound on the total number of iterations is $\sum_{i=0}^{|\Sigma|^k-1} C(i)$.

Algorithm 1 Generate a (C, t, Σ)-joker de Bruijn sequence

```
Set CURR to be an arbitrary k-mer w, s.t. w ∈_t [Σ ∪ {x}]^k.
Initialize SEQ to CURR.
Initialize array A of k-mer counts to 0.
while there is any k-mer y s.t. A(y) < C(y) do
  Initialize MAX to 0.
    for all k-mers w, s.t. w ∈_t [Σ ∪ {x}]^k do
      Set COUNT to the number of new k-mers covered by CURR_{2:k} · w.
      if COUNT > MAX then
        MAX = COUNT.
        MAX_K = w.
      end if
    end for
    Set SEQ = SEQ · MAX_K.
    Update A by the number of new k-mers covered by CURR_{2:k} · MAX_K.
    Set CURR = MAX_K.
end while
Output sequence SEQ.
```

Theorem 1. *The running time of Algorithm 1 is bounded by* $O(\sum_{i=0}^{|\Sigma|^k-1} C(i) \cdot |\Sigma|^{k-|t|_1} \cdot k)$.

Proof. The while loop runs at most $\sum_{i=0}^{|\Sigma|^k-1} C(i)$ iterations by Corollary 1. The inner for loop runs $|\Sigma|^{k-|t|_1}$ iterations since it iterates over all k-mers over $\Sigma \cup \{x\}$ that follow t. Inside the for loop, exactly $2k-1$ k-mers in $CURR_{2:k} \cdot MAX_K$ are examined. We assume that examining each k-mer takes constant time $O(1)$ as it is one array operation. Thus, the total running time is $O(\sum_{i=0}^{|\Sigma|^k-1} C(i) \cdot |\Sigma|^{k-|t|_1} \cdot k)$.

3.2. ILP formulation

Next, we present a novel ILP formulation to solve the MINIMUM-LENGTH (C, t, Σ)-JOKER DE BRUIJN problem. We start by defining variables. Y variables are k-mer counts of k-mers that include $|t|_1$ joker characters. There are $k \cdot |\Sigma|^{k-|t|_1}$ integer variables $Y_{i,j}$. Each $Y_{i,j}$ corresponds to the number of times a k-mer with $|t|_1$ joker characters at positions following cyclic shift of offset j of template t and the rest of the positions as $(k-|t|_1)$-mer i occurs in the sequence. For simplicity, we solve the problem of generating a cyclic sequence, but it can be easily turned into a linear sequence by a modification similar to that presented by D'Addario et al. (2012).

As we aim for the shortest sequence, the objective function is

$$\min \sum_{i=0}^{|\Sigma|^{k-|t|_1}-1} \sum_{j=1}^{k} Y_{i,j} \tag{1}$$

The first constraint is the coverage constraint, which requires that all k-mers occur as the number of times according to C. Let $f(i, j)$ be the $(k-|t|_1)$-mer of all positions, but the joker positions of cyclic shift j of template t of k-mer i are

$$\sum_{j=1}^{j=k} Y_{f(i,j),j} \geq C(i) \qquad 1 \leq i \leq |\Sigma|^k \tag{2}$$

The second constraint guarantees that k-mer occurrences can form a (cyclic) sequence. We require that for each $(k-1)$-mer, the number of k-mers with that $(k-1)$-mer in their suffix is equal to the number of k-

mers with that $(k-1)$-mer in their prefix. Denote $p_x(i)$ and $s_x(i)$ as the x-long prefix and suffix of i, respectively.

$$\sum_{s_{k-|t|_1-1}(i')=i} Y_{i',j+1} = \sum_{p_{k-|t|_1-1}(i')=i} Y_{i',j} \qquad 1 \le j \le k-1$$
$$i \in \{p_{k-1}(i) = w | \forall_{t' \text{ cyclic shift of t}} \forall w \in_{t'} [\Sigma \cup \{x\}]^k\} \tag{3}$$

3.3. RC *covering all* k *-mers*

To further shrink libraries over double-stranded DNA, we utilize the reverse complement property and generate a (C, t, Σ)-RC-joker de Bruijn sequence. We made two modifications to the algorithms above. For Algorithm 1, whenever we consider and choose a new addition of k characters (lines 7 and 14), we need to account for both the k-mers and their reverse complement. For the ILP formulation, we modified the coverage constraint (Eqn. 2). The modified constraint is

$$\sum_{j=1}^{j=k} Y_{f(i,j),j} + Y_{f(RC(i),j),j} \ge C(i) + C(RC(i)) \qquad 1 \le i \le |\Sigma|^k \tag{4}$$

4. RESULTS

4.1. Hardness result

Given a set of strings $s_1, \ldots, s_n$, the shortest common superstring (SCS) optimization problem is to find the shortest string S such that all s_i are substrings of S. SCS was proven to be NP-hard (Räihä and Ukkonen, 1981). We consider a problem equivalent to finding the SCS of a set of strings of length k, with alphabet size $\Sigma \ge 4$, while allowing the superstring to have joker/wildcard characters at fixed positions every k characters. Here, we show that adding a single wildcard no more than once every k characters to a superstring where all substrings are of length k remains NP-hard.

Theorem 2. *MINIMUM-LENGTH* (C, t, Σ)*-JOKER DE BRUIJN SEQUENCE is NP-hard.*

Proof. We build on a reduction from $O(1)$-degree Vertex Cover for the hardness proof (Vassilevska, 2005). Given an instance to Vertex Cover $G=(V,E)$ with $|V|=n$ and $|E|=m$, we construct unique labels over $\Sigma=\{0, 1, 2, 3\}$ for each vertex that are greater than Hamming distance 1 from any other label. That way, joker characters cannot be used to cover multiple labels. Each vertex is assigned a unique binary string over {0,1} of length $\lceil \log_2 n \rceil$. Denote the string as s_a for vertex a. Then, let the string $23s_as_a32$ be an encoding of the vertex labels of length $4+2\lceil \log_2 n \rceil$. Since the unique string is doubled, changing any single character in the string will not give another valid vertex label encoding. In addition, the sentinel characters, 23 and 32, at the ends, which are not used within the body of the label, prevent two labels from overlapping by more than the single character 2 even when jokers are allowed. Let an edge (a, b) be represented by strings *abab* and *baba* (merging adjacent 2s as possible). This is a set of strings of equal length $k=13+8\lceil \log_2 n \rceil$, corresponding to the k-mer size. The set of strings representing the set of edges is the input of the Joker De Bruijn problem, allowing one joker character per k characters.

$\rightarrow$ Suppose G has a covering vertex set S of size κ. Assign every edge (a, b) to its covering vertex (or arbitrarily if both vertices are in S). If a is the assigned vertex for the edge (a, b), overlap the two strings to get *ababa*, else overlap them the other way to get *babab*. Then, for every vertex $c \in S$, we can overlap all assigned edge strings by 1 to get $ca_1ca_1ca_2ca_2c \ldots ca_{\kappa_c}ca_{\kappa_c}c$ of length $4\kappa_c+1$ labels, where κ_c is the number of edges assigned to vertex $c \in S$. By concatenating all such strings together, we get a superstring of length $4m+\kappa$ labels.

$\leftarrow$ Conversely, it can be shown that all sequences for the Joker De Bruijn problem can be reduced by reordering and overlapping to have a length of $4m+\kappa$ labels, which can be translated in polynomial time to a vertex cover. Thus, if we can get a joker de Bruijn sequence of length $4m+\kappa$ labels, we can get a vertex cover of size $\le \kappa$.

Making use of exact bounds from the $O(1)$-degree Vertex Cover problem, it is possible to show that SCS is APX-hard and, by label construction earlier, the Minimum-Length Joker de Bruijn problem with one

joker character per k characters is also APX-hard and thus NP-hard. Note that the same proof holds with minor modifications for any bounded number of joker characters per k.

4.2. Implementation

We implemented the algorithms in Java. We used Gurobi ILP solver, version 6.5.2 (Gurobi Optimization, 2015). We set the method parameter in Gurobi to 3, as recommended, to improve the running time of the root relaxation process. We set a time limit for the ILP solver since solutions for $k \geq 5$ for DNA and $k \geq 3$ for an amino acid alphabet covering all k-mers with template t of weight 1 did not terminate based on the default criteria. Running times were benchmarked on a single CPU of a 20-CPU Intel Xeon E5-2650 (2.3 GHz) machine with 384 GB 2133 MHz RAM.

4.3. Theoretical lower bound

We prove theoretical lower bounds for the (C, t, Σ)-de Bruijn sequence.

Theorem 3. *Denote* $n(C, t, \Sigma)$ as *the length of a* (C, t, Σ)*-de Bruijn sequence Then,*

$$n(C, t, \Sigma) \geq \sum_{i=0}^{|\Sigma|^k - 1} C(i) / |\Sigma|^{|t|_1} \tag{5}$$

Proof. The number of k-mers is $\sum_{i=0}^{|\Sigma|^k - 1} C(i)$. Since there are exactly $|t|_1$ joker characters per k-mer, the number of k-mers in the sequence can be reduced by at most $|\Sigma|^{|t|_1}$. For a noncyclic sequence, $k-1$ characters need to be added.

4.4. Results of greedy heuristic and ILP solver

We ran the greedy heuristic on $5 \leq k \leq 8$ for a DNA alphabet, with and without the reverse complement feature, and $3 \leq k \leq 4$ for an amino acid alphabet, with $C = 1^{|\Sigma|^k}$ and $t = 0^{k-1}1$. We then ran the ILP solver, starting from the greedy solution, with a time limit of 4 weeks. We compared the solution with a random addition of k-mers that follow t and the original de Bruijn sequences without joker characters. Results are summarized in Table 2.

To test the performance in covering k-mers multiple times, we ran the greedy heuristic on $k = 6$, DNA alphabet, $t = 0^{k-1}1$ and $C = p^{|\Sigma|^k}$, where $1 \leq p \leq 16$. We compared the results with the original de Bruijn sequence and a theoretical lower bound. Results are summarized in Table 3.

5. DISCUSSION

Sequence libraries that cover all k-mers are instrumental in measuring protein interactions in a universal and unbiased manner, but they are limited by the exponential growth of k-mers as k increases. Shrinking

Table 2. Results of Greedy Heuristic and Integer Linear Programming Solver in Generating $(1^{|\Sigma|^k}, 0^{k-1}1, \Sigma)$-Joker de Bruijn and $(1^{|\Sigma|^k}, 0^{k-1}1, \Sigma)$-RC-Joker de Bruijn Sequences

Alphabet	*DNA*				*Reverse complement DNA*				*Amino acid*	
K	*5*	*6*	*7*	*8*	*5*	*6*	*7*	*8*	*3*	*4*
de Bruijn	1028	4101	16,390	65,543	516	2085	8198	32,903	8002	160,003
Random	1544	7259	33,438	153,447	719	3293	16,127	70,007	2426	69,483
Greedy	314	1415	4689	19,903	204	797	2519	10,983	629	9919
ILP	290	1159	4436	19,903	162	678	2413	10,784	623	9699
Lower bound	260	1029	4102	16,391	132	525	2054	8231	402	8003
Greedy/de Bruijn	0.31	0.35	0.29	0.30	0.40	0.38	0.31	0.33	0.08	0.06
ILP/de Bruijn	0.28	0.28	0.27	0.30	0.31	0.33	0.29	0.33	0.08	0.06
Greedy time [s]	1.12	7.26	50.78	1213	1.58	9.54	74.96	1647	5.21	1129

Results are compared with the original de Bruijn sequence, a random algorithm, and a theoretical lower bound.
ILP, integer linear programming.

TABLE 3. RESULTS OF GREEDY HEURISTIC ON $\kappa=6$, $t=0^{k-1}1$, $C=p^{|\Sigma|^k}$, WHERE $1 \leq p \leq 16$ AND DNA ALPHABET

P	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*
de Bruijn	4101	8197	12,293	16,389	20,485	24,581	28,677	32,773
Greedy	1415	2435	3491	4625	5537	6503	7661	8681
Lower bound	1029	2053	3077	4101	5125	6149	7173	8197
Greedy/de Bruijn	0.35	0.30	0.28	0.28	0.27	0.26	0.27	0.26
P	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*
de Bruijn	36,869	40,965	45,061	49,157	53,253	57,349	61,445	65,541
Greedy	9791	10,769	11,849	12,857	13,937	15,011	16,139	17,189
Lower bound	9221	10,245	11,269	12,293	13,317	14,341	15,365	16,389
Greedy/de Bruijn	0.27	0.26	0.26	0.26	0.26	0.26	0.26	0.26

Results are compared with the original de Bruijn sequence and a theoretical lower bound.

these k-mer libraries is needed to enable an increase in k to measure interactions in greater detail. In this work, we solved this problem by utilizing a novel idea of using joker characters that represent all possible characters in the alphabet. We presented the first algorithm to solve the problem of covering a given set of k-mers, such that the positions and number of the joker characters follow a given template. We prove that the problem is NP-hard and suggest a novel heuristic to solve it. The solution is based on a greedy heuristic that performs quite well by itself and then shows improvement by solving an ILP formulation. The results are very close to theoretical lower bounds, implying that the solution is near optimal.

One clear advantage of our solution is its generality and flexibility. The alphabet is given as the input, enabling a solution to any set of characters, for example, unnatural amino acids in the amino acid alphabet. Moreover, since the problem is to cover a given set of k-mers, we can support exclusion of specific k-mers for technical reasons. More generally, the solution also supports variable k-mer multiplicities and different positions and numbers of joker characters.

There are several limitations in our study. First, our algorithm is not guaranteed to produce an optimal result in polynomial time. The greedy heuristic is not guaranteed to produce an optimal result. However, we show empirically that it performs very well and produces a result that is close to the lower bound. The ILP solver is guaranteed to produce an optimal result, but is not guaranteed to terminate in polynomial time. In general, the problem we solve, as well as an ILP, is NP-hard. Second, the joker library introduces ambiguity in the measurements. Shrinking the library size comes with a cost of a smaller sample size, lowering the statistical robustness of inferred scores.

Several open questions remain from our study. First, is there an optimal solution that runs in time polynomial in $O(\sum_{i=0}^{|\Sigma|^k-1} C(i))$? Second, is there a good enough heuristic that runs in time linear in the output length, that is, $O(\sum_{i=0}^{|\Sigma|^k-1} C(i)/|\Sigma|^{|t|_1})$, or at least asymptotically faster than Algorithm 1? Third, can we provide tighter lower and upper bounds?

In summary, this work presented a new library design that covers a given set of k-mers at a size that is almost $1/|\Sigma|^{|t|_1}$ smaller compared with current libraries. This implies the ability to measure interactions with longer k-mers and a reduction in cost. We made the implementation and calculated libraries that are freely available for other researchers to use for their sequence sets. With smaller libraries and increase in k, research and measurements of protein interactions will advance significantly.

ACKNOWLEDGMENTS

This work was supported by the National Institutes of Health grant R01GM081871. Y.W.Y. was partially supported by a Hertz Foundation Fellowship.

AUTHOR DISCLOSURE STATEMENT

The authors declare that no competing financial interests exist.

REFERENCES

Berger, M.F., Philippakis, A.A., Qureshi, A.M., et al. 2006. Compact, universal DNA microarrays to comprehensively determine transcription-factor binding site specificities. *Nat. Biotechnol.* 24, 1429–1435.

Blanchet-Sadri, F., Schwartz, J., Stich, S., et al. 2010. Binary de Bruijn partial words with one hole, 128–138. *In* Kratochvíl, J., Li, A., Fiala, J., et al., eds. *Theory and Applications of Models of Computation. TAMC 2010. Lecture Notes in Computer Science, Vol 6108*. Springer, Berlin, Heidelberg.

Chen, H.Z., Kitaev, S., Mütze, T., and Sun, B.Y. 2017. On universal partial words. *Electronic Notes in Discrete Mathematics*, 61, 231–237.

D'Addario, M., Kriege, N., and Rahmann, S. 2012. Designing q-unique DNA sequences with integer linear programs and Euler tours in de Bruijn graphs, 82–93. *In* Böcker, S., Hufsky, F., Scheubert, K., et al., eds. *German Conference on Bioinformatics 2012. OASIcs-OpenAccess Series in Informatics*, Volume 26. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany.

Fordyce, P.M., Gerber, D., Tran, D., et al. 2010. De novo identification and biophysical characterization of transcription-factor binding sites with microfluidic affinity analysis. *Nat. Biotechnol.* 28, 970–975.

Gurard-Levin, Z.A., Kilian, K.A., Kim, J., et al. 2010. Peptide arrays identify isoform-selective substrates for profiling endogenous lysine deacetylase activity. *ACS Chem. Biol.* 5, 863–873.

Gurobi Optimization, U.C. 2018. Gurobi optimizer reference manual. www.gurobi.com

Orenstein, Y., and Berger, B. 2016. Efficient design of compact unstructured RNA libraries covering all *k*-mers. *J. Comput. Biol.* 23, 67–79.

Orenstein, Y., and Shamir, R. 2013. Design of shortest double-stranded DNA sequences covering all *k*-mers with applications to protein-binding microarrays and synthetic enhancers. *Bioinformatics.* 29, i71–i79.

Philippakis, A.A., Qureshi, A.M., Berger, M.F., et al. 2008. Design of compact, universal DNA microarrays for protein binding microarray experiments. *J. Comput. Biol.* 15, 655–665.

Räihä, K.-J., and Ukkonen, E. 1981. The shortest common supersequence problem over binary alphabet is NP-complete. *Theor. Comput. Sci.* 16, 187–198.

Ray, D., Kazan, H., Cook, K.B., et al. 2013. A compendium of RNA-binding motifs for decoding gene regulation. *Nature.* 499, 172–177.

Smith, R.P., Riesenfeld, S.J., Holloway, A.K., et al. 2013. A compact, in vivo screen of all 6-mers reveals drivers of tissue-specific expression and guides synthetic regulatory element design. *Genome Biol.* 14, 1.

Vassilevska, V. 2005. Explicit inapproximability bounds for the shortest superstring problem, 793–800. *In* Jędrzejowicz, J., and Szepietowski, A., eds. *Mathematical Foundations of Computer Science 2005. MFCS 2005. Lecture Notes in Computer Science, Vol 3618*. Springer, Berlin, Heidelberg.

Wyatt, B.J. 2013. *De Bruijn Partial Words*. University of North Carolina at Greensboro.

Address correspondence to:
Prof. Bonnie Berger
Department of Mathematics
Massachusetts Institute of Technology
77 Mass Avenue, 2-373
Cambridge, MA 02139

E-mail: bab@mit.edu