A Uniform Projection Method for Motif Discovery in DNA Sequences

Benjamin Raphael, Lung-Tien Liu, and George Varghese

Abstract—Buhler and Tompa [5] introduced the random projection algorithm for the motif discovery problem and demonstrated that this algorithm performs well on both simulated and biological samples. We describe a modification of the random projection algorithm, called the *uniform projection algorithm*, which utilizes a different choice of projections. We replace the random selection of projections by a greedy heuristic that approximately equalizes the coverage of the projections. We show that this change in selection of projections leads to improved performance on motif discovery problems. Furthermore, the uniform projection algorithm is directly applicable to other problems where the random projection algorithm has been used, including comparison of protein sequence databases.

Index Terms—Motif discovery, transcription factor binding sites, random projection, combinatorial designs, low-discrepancy sequences.

1 Introduction

The problem of discovering signals in a set of DNA sequences (the *motif discovery problem*) is well-studied in computational biology. An important application is to discover binding sites for transcription factors (DNA binding proteins) in a collection of DNA sequences several hundred to several thousand nucleotides in length. These binding sites are frequently short (6-20 nucleotides in length) and not completely conserved. That is, transcription factor binding sites are subject to mutation and, consequently, cannot be identified by pattern matching algorithms that seek exact matches. Since finding mutated signals is a notoriously difficult problem, many algorithms have been proposed for the motif discovery problem. These algorithms employ either a profile or pattern representation of a motif. In profile-based methods such GibbsDNA [15], MEME [1], and CONSENSUS [12], a motif is represented by a position-specific scoring matrix. In pattern-based methods such as TEIRESIAS [19], WINNOWER and SP-STAR [18], MITRA [9], MULTIPROFILER [14], and the algorithms of Marsan and Sagot [16], the motif is represented as a string with mismatches. Numerous other algorithms have been developed for the motif discovery problem and the ever increasing quantity of biological sequence data coupled with high-throughput measurements of gene expression and protein-DNA binding continue to stoke interest in the development, evaluation, and application of motif discovery algorithms.

One motif discovery algorithm that has generated significant interest is random projection algorithm of Buhler and Tompa [5]. The random projection algorithm is a pattern-based method that applies locality-sensitive hashing [13] to search for conserved patterns in a set of sequences. Beyond its application to motif discovery with a pattern motif model, the random projection algorithm was used for motif discovery with a Bayesian network motif model [2] and also applied to comparison of protein sequence databases [4], [3], [11].

In this paper, we describe a modification of the random projection algorithm, called the *uniform projection algorithm*. The uniform projection algorithm replaces the random choice of projections by a greedy heuristic that approximately equalizes the coverage of the chosen projections. In the terminology of optimization theory, the random projection algorithm is a Monte Carlo method for finding a global optimum by applying a local optimization routine to randomly sampled points. In contrast, the uniform projection algorithm biases the selection of sample points to achieve more balanced, or uniform, coverage of the domain. We compare the uniform projection algorithm to the random projection algorithm, and show that on average, the uniform projection algorithm achieves the same rate of success on simulated motif discovery problems as the random projection algorithm with 20-50 percent fewer projections. The reduction in the number of projections translates into a higher rate of success on simulated data when we fix the maximum number of uniform projections to the same value used by Buhler and Tompa in their random projection algorithm. Our uniform projection algorithm can be directly substituted for the random projection algorithm in any problem where the random projection algorithm has proven useful. For clarity of exposition, we will limit the discussion in this paper to the motif discovery problem.

Throughout the paper, we will use the following computational formulation of the motif discovery problem described by Pevzner and Sze [18]. Suppose there is an unknown string M of length l (the *motif*). Given a collection $\mathcal{S} = \{S_1, S_2, \ldots, S_t\}$ of t DNA sequences of fixed length n such that each sequence S_i contains an l-mer M_i, a mutated variant of M, with mutations at d positions, the planted (l, d) *motif discovery problem* is to find M. The major challenge that arises in this formulation of the motif discovery problem is that two motif instances M_i and M_j can be quite different and, thus, direct comparison of the l-mers present in the collection $\mathcal{S}$ is often ineffective. Pevzner and Sze proposed the particularly challenging $(15, 4)$ motif problem in which two instances M_i and M_j of the motif can differ from each other in eight of 15 locations, even though each instance of the motif differs from M in at most four positions.

The (l, d) motif discovery problem models the fact that biological motifs are often conserved at particular positions, but that these positions may not be contiguous. Furthermore, the conserved positions may not be identical over all instances of the motif present in the data. However, one reasonable assumption is that a significant fraction of the motif instances will agree at certain positions. For example, for an (l, d) problem, we expect that several instances M_i agree at k fixed positions, for some integer $k < l - d$. This assumption forms the guiding principle of the random projection algorithm of Buhler and Tompa [5].

2 Random Projection Algorithm

The idea of the random projection algorithm for the (l, d) motif discovery problem is to form a candidate motif model from l-mers in the data $\mathcal{S}$ that agree at k particular positions, for some $k \leq l$. Given a substring $S = s_1 s_2 \ldots s_l$ from the collection $\mathcal{S}$ and k distinct integers $1 \leq p_1 < p_2 < \cdots < p_k \leq l$, we define a (k/l)-*projection* $P(S)$ to be the string $s_{p_1} s_{p_2} \ldots s_{p_k}$. We write the projection operator P by listing the k-tuple of positions: $P = (p_1, p_2, \ldots, p_k)$. We will refer to P simply as a k-projection when the length l is clear from the context.

A single iteration of the random projection algorithm proceeds as follows: Fix a projection length k and choose a projection P_1 by selecting k distinct positions $1 \leq p_1 < p_2 < \cdots < p_k \leq l$ uniformly at random. Create a hash table $\mathcal{H}$ of size 4^k, indexed by all possible DNA strings of length k. Each entry of the hash table consists of a *bucket* where substrings of length l are stored. For each substring S of length l from the collection $\mathcal{S}$, place S into the bucket with index $P_1(S)$. Fix an integer $s > 0$ and call a bucket containing s or more strings an *enriched bucket*. Under the assumption that many instances of a motif agree at some positions then a projection P

- *B. Raphael and G. Varghese are with the Department of Computer Science and Engineering, University of California, San Diego, La Jolla, CA 92093-0114. E-mail: braphael@ucsd.edu, varghese@cs.ucsd.edu.*
- *L.-T. Liu can be reached at E-mail: lungtien_liu@yahoo.com.*

Manuscript received 27 Feb. 2004; revised 21 July 2004; accepted 30 Aug. 2004.
For information on obtaining reprints of this article, please send e-mail to: tcbb@computer.org, and reference IEEECS Log Number TCBB-0027-0204.

onto these positions will result in an enriched bucket. Take the strings in an enriched bucket B as an initial model for the motif. For example, we can use the consensus of the strings in B as an initial model for M or, alternatively, we can form a profile from the strings in B. The initial motif model is then *refined* using a local optimization scheme. Buhler and Tompa [5] employ the expectation maximization (EM) algorithm of Bailey and Elkan [1] as well as a variation of the SP-STAR algorithm of Pevzner and Sze [18] to optimize the initial motif model obtained from an enriched bucket. Thus, the random projection algorithm can be viewed as a procedure for determining good starting points for a local optimization routine such as EM.

We refine each enriched bucket and retain the "best" motif discovered during this process, where "best" is determined by an appropriate motif scoring function. We start a new iteration by choosing a new projection P_2 and repeating the hashing and refinement steps using the new projection P_2. After a suitable number of iterations (choose projection, hash, refine), return the best motif found over all iterations.

The random projection algorithm depends on several parameters whose values must be chosen. For a given (l, d) motif problem we must specify the projection size k, the threshold s for defining an enriched bucket and the number of iterations m of the algorithm. Certainly, we can perform all possible k-projections, but this approach leads to a large number of iterations and, consequently, lengthy running times for the algorithm. For example, for the $(15, 4)$ problem with projection size $k = 7$, there are $\binom{15}{7} = 6,435$ possible projections. For a (l, d) motif problem with fixed projection size k and bucket threshold s, Buhler and Tompa compute the number m_{BT} of random projections to perform in the following way. Let P denote the chosen random projection and define $P(M)$ to be the *planted bucket*. Buhler and Tompa define m_{BT} to be the number of iterations necessary such that there is a 95 percent probability that the planted bucket is enriched on at least one iteration, i.e., s or more motif instances will appear in the same enriched bucket on at least one iteration. For example, for the $(15, 4)$ motif discovery problem, Buhler and Tompa perform a maximum of 172 randomly chosen projections.

3 Uniform Projection Algorithm

Buhler and Tompa employ random projections in order to find good starting points for expectation maximization, a local optimization procedure. They note that on occasion, the random projection algorithm recovers the motif from a bucket other than the planted bucket, an occurrence that they refer to as an "added bonus" of the algorithm [5]. However, this feature highlights another way of viewing the success of the random projection algorithm: the random projection algorithm samples the space of all possible projections, occasionally (and with high probability if at least m_{BT} random projections are performed) finding a "good" projection that produces an enriched bucket that is refined to produce the motif M. From this perspective, sampling the space of all k-projections by random selection is not a very efficient strategy. For example, for the $(15, 4)$ motif problem with projection size 7, if one chooses projections randomly, then, on average, 60,146 projections must be chosen before all $6,435(7/15)$-projections are sampled, a result that follows from the classic *coupon collector problem* [10].

Instead of choosing projections by selecting k positions uniformly at random, we want to bias the choice of projections to sample the space of projections more efficiently. Our motivation is similar to the use of quasirandom or low-discrepancy sequences in Monte Carlo integration and global optimization [17]. In these applications, a relatively small number of carefully chosen sample points provide better performance than randomly chosen points. We define the *discrepancy* between a sequence of projections as follows: First, we represent a (k/l)-projection P as an l-bit binary string with k ones at the positions in P. We define the distance $\delta(P_i, P_j)$ between projections P_i and P_j to be the Hamming distance between their binary representations. The discrepancy d between a sequence $P_1, \ldots, P_M$ of projections is equal to $\max_P \min_{1 \le i \le M} \delta(P, P_i)$, where the maximum is taken over all (k/l)-projections P. Our goal is to find a relatively small sequence of projections with low discrepancy. While the theory and construction of low-discrepancy sequences in $[0, 1]^n$ are well developed, these techniques are not directly applicable in Hamming space. Thus, we investigate a heuristic method to generate sequences of projections with lower discrepancy than sequences of random projections.

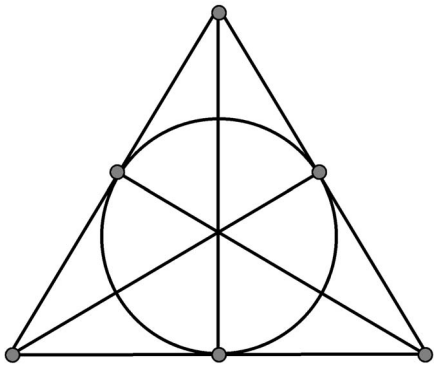

Fig. 1. A 2-(7, 3, 1) combinatorial design, known as the *Fano plane*. The line segments and circle represent the blocks. Each block contains three points and each point contained in exactly two blocks.

An obvious way to reduce the number of projections without changing the discrepancy of the sequence of projections is to choose a new projection P_{M+1} only if it is distinct from the projections $P_1, P_2, \ldots, P_M$ that have already been chosen. Generalizing this observation, we now describe a different method of projection selection that maintains good performance on motif discovery problems while using a smaller number of projections than random projection.

We say that a k-projection P *covers* a j-tuple $J = (n_1, \ldots, n_j)$ if $J \subseteq P$. Fix a motif length l, a projection size $k \le l$, a covering length $j \le k$, and a coverage parameter $\lambda \ge 1$. We desire a solution to the following problem.

Uniform Projection Problem. *Find a sequence of* (k/l)-projections $P_1, P_2, \ldots, P_N$ *such that each* j-tuple $J = (n_1, \ldots, n_j)$ *of positions is covered by* λ *projections.*

A solution to the Uniform Projection Problem gives a sequence of projections with discrepancy $d \le 2(k - j)$; thus, larger values of j will yield sequences with lower discrepancy. For certain values of l, k, j, and λ, the Uniform Projection Problem has a solution. The study of these cases is the subject of *combinatorial design theory* [21], [7], [20], where in the terminology of combinatorial designs, the "points" are the j-tuples and the "blocks" are the k-projections. The sequence of projections is given by a $j - (l, k, \lambda)$ *design* [20] (Fig. 1). Unfortunately, the values of l, k, and j encountered in motif discovery problems rarely coincide to the cases where a solution is given by a combinatorial design. Furthermore, even when a combinatorial design exists for particular values l, k, j, and λ, constructing the design is a difficult problem [7].

For the motif discovery problem, we focus on the case $\lambda = 1$, i.e., each j-tuple is covered once. Furthermore, we content ourselves with an approximate solution to the Uniform Projection Problem, where each j-tuple is covered by *at least* $\lambda = 1$ projections. Thus, we view the Uniform Projection Problem as an instance of the set covering problem [8], where the "points" are the tuples J and the "sets" are the tuples covered by a projection P. We use a

TABLE 1
Average Number of Uniform/Random 7-Projections Required to Cover j-tuples, Where the Average Is Obtained over 20 Runs of the Algorithm

	l					
j	14	15	16	17	18	19
1	2.0/5.3	3.0/5.8	3.0/7.2	3.0/6.4	3.0/7.7	3.0/8.3
2	8.3/17.7	9.2/24.2	10.9/28.4	12.1/31.1	13.8/37.7	15.4/42.3
3	24.7/64.5	31.7/84.3	39.0/94.3	48.3/132.9	58.9/161.3	71.4/189.4
4	74.8/191.3	105.6/289.3	143.9/427.5	192.0/558.6	249.9/707.9	321.1/1022.9
5	258.8/684.1	399.3/1122.0	591.0/1744.1	845.9/2541.4	1185.8/3618.9	1624.1/5399.4
6	914.6/2180.6	1532.7/3933.7	2467.5/6975.8	3842.8/11865.7	5783.1/18234.7	8484.9/27765.6

Approximately 2-3x more random projections are required to cover j-tuples.

greedy approach to the set covering problem similar to the greedy heuristic described by Chváital [6].

In the greedy algorithm, we iteratively choose projections, where at each iteration we build a new k-projection from infrequently covered j-tuples as follows: We maintain a hash table $\mathcal{C}$ indexed by all j-tuples of positions: The entry $\mathcal{C}(J)$ in the hash table for the j-tuple $J = (n_1, n_2, \ldots, n_j)$ is the count $|\{i : J \subseteq P_i\}_{1 \le i \le M}|$ of the currently chosen projections $P_1, P_2, \ldots, P_M$ that cover J. We add a new projection P_{M+1} to our sequence in a greedy fashion by selecting the positions of P_{M+1} to be a union of j-tuples with the lowest counts in $\mathcal{C}$. Often many j-tuples J will have the smallest count in the hash table $\mathcal{C}$ and, so, we randomize the sequence of j-tuples with the same lowest count and select the first j-tuples in the randomized sequence whose union is at least k positions. We randomly drop positions from the last j-tuple in the selected set to obtain exactly k positions.

For example, say the motif length $l = 15$, the projection size $k = 7$ and $j = 4$. There are $\binom{15}{7} = 6,435$ (7/15)-projections and each projection covers 35 4-tuples. The hash table $\mathcal{C}$ is indexed by the $\binom{15}{4} = 1,365$ possible 4-tuples. If $\mathcal{C}((1,5,8,10)) = 0$ and $\mathcal{C}((4,5,9,15)) = 0$, then we can cover these 4-tuples by forming the (7/15)-projection $P = (1,4,5,8,9,10,15)$. Having done so, we now set $\mathcal{C}((1,5,8,10)) = 1$, $\mathcal{C}((4,5,9,15)) = 1$, and we also increase the counts in the hash table of the other 33 subsets of $\{1,4,5,8,9,10,15\}$ of size 4.

In our implementation of the uniform projection algorithm, we construct the full sequence of projections $P_1, P_2, \ldots, P_N$ in batches as follows: First, we construct projections $P_1, P_2, \ldots, P_{M_1}$ which cover all 1-tuples. Next, we greedily augment this sequence of projections with new projections $P_{M_1+1}, P_{M_1+2}, \ldots, P_{M_2}$ such that the augmented sequence $P_1, P_2, \ldots, P_{M_2}$ cover all 2-tuples. We continue augmenting the sequence of projections in this way until all j-tuples are covered. During this process, a new projection P_{M+1} is added to the sequence only if it is distinct from $P_1, P_2, \ldots, P_M$. With this approach, the parameter j does not need to be chosen a priori. Instead, one chooses m, the number of projections to perform (i.e., how long to run the algorithm), with the guarantee that as m increases, all j-tuples are covered for increasing values of j.

4 Results

We wrote implementations of the random projection and uniform projection algorithms in C, which are available at http://www-cse.ucsd.edu/groups/bioinformatics/software.html. To verify our code, we also used Jeremy Buhler's C++ implementation of the random projection algorithm [4] and modified the projection generation routines to employ our uniform projection scheme. Both implementations produced the same results.

We first test if the greedy heuristic for uniform choice of projections provides superior coverage of j-tuples of positions compared to a random choice of projections. We fix the projection size $k = 7$. For fixed values of l and j, we find the average number of projections required to cover j-tuples once, where the average is obtained over 20 runs of the algorithm (Table 1). In general, the number of uniform projections required to cover j-tuples is less than one-half to one-third the number of random projections. A lower bound on the number of projections required to achieve λ-fold coverage of j-tuples is

$$\lambda \frac{\binom{l}{j}}{\binom{k}{j}}.$$

This bound is achieved by a a $j - (l, k, \lambda)$ design, when such a design exists. However, this lower bound is not strict if no design exists. For the case $k = 7$, the greedy algorithm requires roughly twice the number of projections to achieve 6-tuple coverage as an exact design would require. The better coverage of j-tuples achieved by the uniform projection algorithm also results in uniform projection sequences having lower discrepancy than random projection sequences (Fig. 2).

To determine if the greater j-tuple coverage and lower discrepancy of uniform projections lead to better results in motif discovery, we generated planted (l, d) motif discovery problems using the following motif model (FM model) of Pevzner and Sze [18]. Select a motif M by choosing the l letters in M randomly from the set $\{\mathrm{A}, \mathrm{C}, \mathrm{T}, \mathrm{G}\}$. Generate an instance M_i of the motif by selecting d positions of M randomly and mutating the letters at these positions. Create a DNA sequence S_i of length n, by first inserting the motif instance M_i at a random location in S_i and then generating the remaining letters of the sequence S_i randomly from the set $\{\mathrm{A}, \mathrm{C}, \mathrm{T}, \mathrm{G}\}$. All random choices are performed from the uniform distribution. Repeat this procedure for $i = 1, \ldots, t$ to generate t DNA sequences of length n, each with a mutated variant M_i of the motif M.

Table 2 shows a comparison of uniform versus random projections in different (l, d) motif discovery problems with $t = 20$ sequences of length $n = 600$ nucleotides, projection size $k = 7$, and

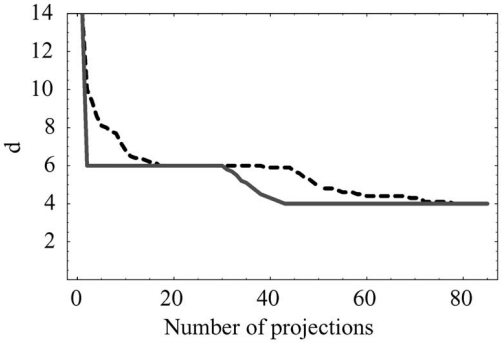

Fig. 2. The discrepancy d of a sequence of uniform $(7/14)$-projections (solid gray curve) is lower than a sequence of random $(7/14)$-projections (dashed curve), demonstrating that uniform projections provide more efficient sampling of the space of projections. Each curve is an average of 20 projection sequences.

TABLE 2
Comparison of Uniform versus Random Selection of Projections on Simulated (l, d) Motif Discovery Problems with $t = 20$ Sequences of Length $n = 600$, Projection Length $k = 7$, and Bucket Threshold $s = 4$

			success	average number projections		
l	d	m_{max}	rate	random (R)	uniform (U)	U/R
12	3	792	1	71.4	38.1	0.53
13	3	1716	1	5.7	2.9	0.50
14	4	3432	1	309.8	187.2	0.60
15	4	6435	1	3.9	4.1	1.05
16	5	11440	1	606.1	471.1	0.78
17	5	19448	1	9.7	12.2	1.26
18	6	31824	1	3702.7	2824.2	0.76
19	6	50388	1	91.7	55.5	0.60

In each case, the average is taken over 20 replicates. The success rate is defined as $1 - \frac{\text{num. failures}}{20}$, where a failure means that the motif consensus does not agree with the consensus of the planted motifs after $m_{max} = \binom{l}{7}$ projections.

bucket threshold $s = 4$. For each value of (l, d) the results of 20 different randomly generated problems are shown. A failure means that the recovered motif consensus does not agree with the consensus of the 20 planted motif instances, after performing all $m = \binom{l}{k}$ possible k-projections and the *success rate* is defined as $1 - \frac{\text{num. failures}}{20}$. With the exception of a few simple problems—where the motif is found after a relatively small number of projections—fewer uniform projections are required to find the motif. For these more difficult problems, the reduction in the number of projections is more than 20 percent.

In the planted (l, d) motif discovery, the motif M is known and, thus, we can run the algorithm until either M is found or all projections have been performed. In a real biological motif discovery problem, the motif M is unknown, and thus we must choose a fixed number of projections to perform. For a given motif length l, projection size k, and bucket threshold s, Buhler and Tompa compute the number of random projections, m_{BT}, necessary such that there is a 95 percent probability that s or more motif instances appear in an enriched bucket during at least one iteration. When we perform at most m_{BT} projections, we see that the success rate of both the random projection and uniform projection algorithms fall below one (Table 3). For the difficult $(14, 4)$, $(16, 5)$, and $(18, 6)$ motif problems, the uniform projection algorithm finds the planted motif in 10 percent more cases than the random projection algorithm. When both algorithms succeed, the uniform projection algorithm requires 20-50 percent fewer projections. Performance improvements are similar on sample biological datasets. A rigorous analysis of a stopping criterion for the uniform projection algorithm is difficult. Nevertheless, our results indicate that at most $0.8\, m_{BT}$ uniform projections are sufficient to obtain similar success rates as random projections; alternatively, m_{BT} uniform projections are generally more successful than the same number of random projections.

These computational experiments reveal that in many cases the uniform projection algorithm performs better than the random projection algorithm, either by finding the motif with fewer projections (i.e., faster), or by finding the motif in cases where random projection fails. The costs in time and memory for the creation of uniform projections is negligible and the reduction in running times using uniform projection versus random projection is approximately equal to the reduction in the number of required projections (20-50 percent), as the rate limiting step of both algorithms is the EM refinement of enriched buckets.

The uniform projection algorithm can be applied to other problems where random projection has been useful, including comparison of protein databases [3], [11]. Further study of low-discrepancy sequences in Hamming space may result in methods for generating even more efficient sequences of uniform projections, extending the utility of the algorithm in bioinformatics applications.

TABLE 3
Uniform versus Random Projections Maximum of m_{BT} Projections Allowed

			success rate		average number projections		
l	d	m_{BT}	random	uniform	random R	uniform U	U/R
12	3	259	1	1	71.4	38.1	0.53
13	3	62	1	1	5.7	2.9	0.50
14	4	647	0.85	0.95	244.8	176.6	0.72
15	4	172	1	1	3.9	4.1	1.05
16	5	1292	0.85	0.9	433.7	423.8	0.98
17	5	378	1	1	9.7	12.2	1.26
18	6	2217	0.7	0.8	1500.6	1598.4	1.07
19	6	711	1	1	91.7	55.5	0.60

The uniform projection algorithm achieves higher rates of success with fewer projections.

ACKNOWLEDGMENTS

The authors are indebted to Jeremy Buhler for providing his implementation of the random projection algorithm. They also thank Pavel Pevzner for helpful discussions in the preparation of this manuscript and an anonymous referee for pointing out the connection to quasirandom sequences. Benjamin Raphael was supported by a fellowship from the Alfred P. Sloan Foundation.

REFERENCES

[1] T.L. Bailey and C. Elkan, "Unsupervised Learning of Multiple Motifs in Biopolymers Using Expectation Maximization," *Machine Learning*, vol. 21, nos. 1-2, pp. 51-80, 1995.
[2] Y. Barash, G. Elidan, N. Friedman, and T. Kaplan, "Modeling Dependencies in Protein-DNA Binding Sites," *Proc. Seventh Int'l Conf. Research in Computational Molecular Biology (RECOMB)*, 2003.
[3] J. Buhler, "Efficient Large-Scale Sequence Comparison by Locality-Sensitive Hashing," *Bioinformatics*, vol. 17, no. 5, pp. 419-428, 2001.
[4] J. Buhler, "Search Algorithms for Biosequences Using Random Projection," PhD thesis, Univ. of Washington, 2001.
[5] J. Buhler and M. Tompa, "Finding Motifs Using Random Projections," *J. Computational Biology*, vol. 9, no. 2, pp. 225-242, 2002.
[6] V. Chváital, "A Greedy Heuristic for the Set-Covering Problem," *Math. Operations Research*, vol. 4, no. 3, pp. 233-235, 1979.
[7] *Algorithms in Combinatorial Design Theory*, C.J. Colbourn and M.J. Colbourn, eds. Amsterdam: North-Holland Publishing, 1985.
[8] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, *Introduction to Algorithms*. Cambridge, Mass.: MIT Press, 1990.
[9] E. Eskin and P.A. Pevzner, "Finding Composite Regulatory Patterns in DNA Sequences," *Bioinformatics*, vol. 18, no. 1, pp. 354-363, 2002.
[10] W. Feller, *An Introduction to Probability Theory*. New York: Wiley, 1971.
[11] E. Halperin, J. Buhler, R. Karp, R. Krauthgamer, and B. Westover, "Detecting Protein Sequence Conservation via Metric Embeddings," *Bioinformatics*, vol. 19, no. 1, pp. 122-122, 2003.
[12] G. Hertz and G. Stormo, "Identifying DNA and Protein Patterns with Statistically Significant Alignments Of Multiple Sequences," *Bioinformatics*, vol. 15, pp. 563-577, 1999.
[13] P. Indyk and R. Motwani, "Approximate Nearest Neighbors: towards Removing the Curse of Dimensionality," *Proc. 13th Ann. ACM Symp. Theory of Computing*, 1998.
[14] U. Keich and P.A. Pevzner, "Finding Motifs in the Twilight Zone," *Bioinformatics*, vol. 18, no. 10, pp. 1374-1381, 2002.
[15] C. Lawrence, S. Altschul, M. Boguski, J. Liu, A. Neuwald, and J. Wootton, "Detecting Subtle Sequence Signals: A Gibbs Sampling Strategy for Multiple Alignment," *Science*, vol. 262, pp. 208-214, 1993.
[16] L. Marsan and M.F. Sagot, "Algorithms for Extracting Structured Motifs Using a Suffix Tree with an Application to Promoter and Regulatory Site Consensus Identification," *J. Computational Biology*, vol. 7, nos. 3-4, pp. 345-362, 2000.
[17] H. Niederreiter, "Random Number Generation and Quasi-Monte Carlo Methods," *Proc. CBMS-NSF Regional Conf. Series in Applied Math.*, vol. 63, 1992.
[18] P.A. Pevzner and S.H. Sze, "Combinatorial Approaches to Finding Subtle Signals in DNA Sequences," *Proc. Eighth Int'l Conf. Intelligent Systems for Molecular Biology*, pp. 269-278, 2000.
[19] I. Rigoutsos and A. Floratos, "Combinatorial Pattern Discovery in Biological Sequences: The TEIRESIAS Algorithm," *Bioinformatics*, vol. 14, no. 1, pp. 55-67, 1998.
[20] D.R. Stinson, *Combinatorial Designs: Constructions and Analysis*. New York: Springer-Verlag, 2004.
[21] J.H. vanLint and R.M. Wilson, *A Course in Combinatorics*. Cambridge Univ. Press, 1992.