Finding Motifs in DNA Sequences Using Low-Dispersion Sequences

XUN WANG¹, YING MIAO¹, MINQUAN CHENG²

ABSTRACT

Motif finding problems, abstracted as the planted (l, d) -motif finding problem, are a major task in molecular biology—finding functioning units and genes. In 2002, the random projection algorithm was introduced to solve the challenging $(15, 4)$ -motif finding problem by using randomly chosen templates. Two years later, a so-called uniform projection algorithm was developed to improve the random projection algorithm by means of low-dispersion sequences generated by coverings. In this article, we introduce an improved projection algorithm called the low-dispersion projection algorithm, which uses low-dispersion sequences generated by developed almost difference families. Compared with the random projection algorithm, the low-dispersion projection algorithm can solve the (l, d) -motif finding problem with fewer templates without decreasing the success rate.

Key words: developed almost difference family, low-dispersion sequence, motif finding, random projection, uniform projection.

1. INTRODUCTION

IN MOLECULAR BIOLOGY, GENES ARE BASIC FUNCTIONING units containing genetic information and can be used as templates for protein transcription. The process of protein transcription begins with binding transcription factor protein to a binding site (a DNA segment) on genomic sequence. DNA segments that act as binding sites are called motifs. The motif finding problem, usually known as the planted (l, d) -motif finding problem, is a fundamental problem in molecular biology with important applications in locating regulatory sites and drug target identification. The problem abstracts the task of discovering binding sites for transcription factors in a collection of DNA sequences. 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 seeking exact matches. In laboratories, a number of experimental motif detecting methods have been developed. The interested reader is referred to Li and Tompa (2006) and Das and Dai (2007) for surveys. Among them, DNase footprinting (Galas and Schmitz, 1978) and gel shift assay (Garner and Revzin, 1981) are two famous ones that can achieve significant accuracy rates in detecting motifs in DNA sequences and genomes. In recent years, with the development of the high-throughput sequencing approaches, even a single experiment can generate a huge number of DNA sequence data. In this condition, detecting motifs by experimental methods will become labor-intensive, time

¹Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba, Ibaraki, Japan.

²Department of Mathematics, Guangxi Normal University, Guilin, P.R. China.

consuming, and expensive. A feasible way to solve this problem is using computational approaches to detect unknown motifs and their locations in genomes, and then verifying the detected motifs by experimental methods. With this purpose, the motif-detecting problem was abstracted and formally reformulated as the planted (l, d) -motif finding problem (Pevzner and Sze, 2000).

Planted (l, d) -Motif Finding Problem: Let M be a fixed but unknown nucleotide sequence of length l . Suppose that M occurs once in each of h background sequences of common length n , but that each occurrence of M is corrupted by exactly d point substitutions in positions chosen independently at random. Given the h sequences, find the h motifs and recover the consensus M .

We mention the unknown nucleotide sequence M as a consensus, the occurrence of M in each background sequence as a motif. Throughout this article, the discussion of the motif finding problem will build on the following DNA model. Suppose that M is a consensus created by choosing l bases randomly; $S_1, S_2, \dots, S_h$ are randomly generated DNA sequences with common length $n - l$; h motifs are created as a mutated variant of M , with d different positions mutated, and each motif is assigned to a random position of a background sequence, one motif per sequence. We say that the model above has a size of $h \times n$. The model reveals that, firstly, the mutated positions in a motif do not necessarily form a contiguous string, and different motifs can mutate in different positions; secondly, the exact positions where motifs are planted are unknown. These make the direct comparison algorithm often ineffective on the motif finding problem.

There are many local search algorithms designed for the motif finding problem, such as Bailey and Elkan's MEME (1995), Hertz and Stormo's CONSENSUS (1999), and Lawrence et al.'s Gibbs sampler (1993), in which finding motifs can be seen as retrieving short strings in computer science. These algorithms employ heuristic methods based on local search to find motifs that maximize their score functions, such as the likelihood ratio.

Despite many studies, for some particular values of l and d , the motif finding problem is far from being resolved. The following *challenging problem* was raised by Pevzner and Sze (2000).

(15, 4)-Motif Finding Challenging Problem: Find 20 occurrences of motifs of length 15 in 20 background sequences of length 600, where each occurrence of the motif differs from its consensus in four randomly chosen positions.

Pevzner and Sze (2000) showed that local search-based algorithms perform poorly on the (15, 4)-motif finding challenging problem. This is because any two motifs may differ by as many as eight positions. The numerous spurious with 8 mutations in 15 positions disguise the real motif, which makes the (15, 4)-motif finding problem inherently intractable. On the other hand, local search methods usually trap at a local maximum of score functions corresponding to a randomly choosing initial value, missing the planted motif despite its much higher score.

The random projection algorithm was designed by Buhler and Tompa (2002), applying locality-sensitive hashing, to solve the motif finding problem. The algorithm uses randomly chosen templates to hash the same fragments of DNA sequences together. Buhler and Tompa (2002) claimed that the random projection algorithm can solve the (15, 4)-motif finding challenging problem in a few minutes and performs better than most of the existing local search-based algorithms.

Although the random projection algorithm achieved well performances in solving the (l, d) -motif finding problem, the number of templates used in the algorithm is significantly large. Raphael et al. (2004) described a modification of the random projection algorithm, called the uniform projection algorithm. The uniform projection algorithm improves the random projection algorithm by replacing the randomly chosen templates with blocks of some combinatorial structure called covering and achieves the same rate of success as the random projection algorithm with 20 percent fewer templates.

Despite the improvement to the random projection algorithm, the uniform projection algorithm is limited by the inherent vices of coverings. Firstly, coverings may generate repeated templates, which are useless in the projection algorithm. On the other hand, templates with low dispersion do better on projection, but coverings with low dispersion may generate even more templates than those needed in the random projection algorithm. In this work, we propose a further improvement on the uniform projection algorithm, called the low-dispersion projection algorithm. We first show that the low-dispersion projection algorithm can solve the (l, d) -motif finding problem more efficiently without decreasing the rate of success compared with the random projection algorithm, then we show this algorithm can generate templates with low dispersion without repeating, and finally we notice that the number of templates generated can be controlled in an acceptable range.

The remainder of this article is organized as follows. Section 2 recapitulates the random projection algorithm and describes how to compute the bucket threshold s , the only parameter in the random projection algorithm not theoretically determined by Buhler and Tompa (2002). Section 3 sketches the uniform projection algorithm. We describe in detail the low-dispersion projection algorithm in Section 4. The comparison between the low-dispersion projection algorithm and the random projection algorithm will be given in Section 5. Some concluding remarks are listed in Section 6.

2. THE RANDOM PROJECTION ALGORITHM REVISITED

Although the motifs are created and planted in background sequences with uncontrollable factors, one reasonable assumption is that a significant fraction of the motifs will have a subsequence remain unaffected by mutation. For example, for the (l, d) -motif finding problem, it is assumed (Buhler and Tompa, 2002) that some of the h motifs agree at k positions, with some integer $k < l - d$. The essential idea of the random projection algorithm for the (l, d) -motif finding problem is repeatedly choosing k positions out of l uniformly at random, with the hope that some randomly chosen k positions are exactly the unaffected positions of motifs. The l -mers in the background sequences that agree at the k chosen positions will be used as a candidate motif model. This strategy can bypass the complication of the motif finding problem model. We note that the random projection strategy is primarily an initialization technique to improve the sensitivity of local search algorithms. In the random projection algorithm, the motif model formed by random projection is taken as an initial value of the expectation maximization (EM) algorithm, which is a well-known local search algorithm (Buhler and Tompa, 2002).

Given an l -mer $a = a_0a_1 \cdots a_{l-1}$ and a set $t = \{x_1, x_2, \dots, x_k\}$, $0 \leq x_1 < x_2 < \cdots < x_k \leq l-1$, t is said to be an (l, k) -template and $P(a, t) = a_{x_1}a_{x_2} \cdots a_{x_k}$ is defined to be the concatenation of nucleotides from a as defined by the template t .

The random projection algorithm processes as follows, referring to the model of motif finding problem.

- Step 1. For the (l, d) -motif finding model of size $h \times n$ with h planted motifs, choose a template t of size k uniformly at random, $k < l - d$, and create a hash table of size 4^k , labeled by all possible DNA strings of length k .
- Step 2. For every substring a of length l in the motif finding model, project the substring by the chosen template t and record the substring in the bucket of the hash table indexed by $P(a, t)$.
- Step 3. Fix a hash table threshold $s > 0$ and call a bucket of the hash table containing s or more substrings an enriched bucket.
- Step 4. Take the substrings in each enriched bucket as an initial model of motifs. The initial model of motifs is then refined by a local search based algorithm.

The above process will be put into iteration, and after a suitable number of iterations, the “best” motif found over all iterations will be returned, where “best” is determined by an appropriate evaluation function such as likelihood ratio.

The fundamental intuition in Step 3 is that, if $k < l - d$, there is a good chance that at least s planted motifs will hash into the same enriched bucket. If k is not too small, it is unlikely that some other spurious motifs from the background sequences will hash into the enriched bucket, because such spurious motifs must agree with the consensus at all the k chosen positions. Among these enriched buckets, Buhler and Tompa (2002) called the one labeled by $P(M, t)$ a planted bucket. They believed that there is some enriched bucket that is the planted bucket and can be refined to the consensus M .

The refinement algorithm in Step 4, as used by Buhler and Tompa (2002), is the EM algorithm, as formulated for the motif finding problem by Lawrence and Reilly (1990). After the refinement, the best motif will be returned as the outcome of the random projection algorithm under the current iteration.

To fully specify the random projection algorithm, Buhler and Tompa (2002) described how to compute the template size k and the number of iterations m , where

$$m = \left\lceil \frac{\log(1-q)}{\log B_{h,p}(s)} \right\rceil.$$

Here, $p = \frac{\binom{l-d}{k}}{\binom{l}{k}}$ is the probability that each motif occurrence in the model hashes to the planted bucket, $B_{h,p}(s)$ is the probability that there are fewer than s successes in h independent Bernoulli trials with success

probability p , and q is the least probability that at least one of the m iterations produces an enriched bucket containing at least s motif occurrences. However, Buhler and Tompa (2002) were unable to find any theory to determine a lower bound for s , though they assumed s to be 3 or 4 by empirical knowledge. Now we determine a lower bound for s theoretically.

The event that a motif occurrence hashes to the planted bucket follows the binomial distribution $B(h, p)$, which can be approximated by the normal distribution $N(hp, hp(1-p))$. The probability that at least s successes in h independent trials is given by

$$\sum_{i=s}^h \binom{h}{i} p^i (1-p)^{h-i}.$$

The above value can be approximately computed by first transforming the normal distribution to the standard normal distribution $N(0, 1)$ by $x = \frac{s-hp}{\sqrt{hp(1-p)}}$, and then calculating 95 percent of the area under the curve of the probability density function

$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}.$$

By checking the standard normal distribution table, the value of x is approximately 1.645. From the equality

$$\frac{(s-0.5)-hp}{\sqrt{hp(1-p)}} \approx 1.645,$$

we can get $s \approx 3.148$, which results in $s = 3$.

For the (l, d) -motif finding problem in an $h \times n$ DNA model, the efficiency of the random projection algorithm is mostly determined by the number of iterations m , that is, the number of templates. In the random projection algorithm, templates are chosen independently at random. According to the theories of experimental designs (Raghavarao, 1988) and quasi-Monte Carlo methods (Niederreiter, 1992), choosing templates with “balanced” and “low-dispersion” properties will make the algorithm more efficient. The following two sections describe the improvements on the random projection algorithm.

3. UNIFORM PROJECTION PROBLEM REVISITED

The random projection algorithm (Buhler and Tompa, 2002) was introduced to find good starting points for the EM algorithm. Raphael et al. (2004) viewed the success of the random projection algorithm in the following way: the random projection algorithm samples the space of all possible templates, and occasionally finds a “good” template to enrich a bucket that can be refined to produce the consensus M . From this perspective, sampling the space of all templates of size k with random selection is not a very efficient strategy. Instead of choosing templates by selecting k positions out of l uniformly at random, Raphael et al. (2004) suggested a strategy that biases the choice of templates to sample the space of templates more efficiently. Similar to the application of the low-dispersion sequences in Monte Carlo integration and global optimization (Niederreiter, 1992), Raphael et al. (2004) believed that a relatively small number of carefully chosen templates with low dispersion will provide better performance than randomly chosen templates do.

Let an (l, k) -template t be represented by an l -bit binary string with k 1's corresponding to the positions in t . The distance between two templates t_i and t_j , $\delta(t_i, t_j)$, is Hamming distance between their binary representations. The dispersion $d_N(T; \mathfrak{T})$ of templates $T = \{t_1, t_2, \dots, t_N\}$ in the template space $\mathfrak{T}$ is defined to be $\max_{t \in \mathfrak{T}} \min_{1 \leq i \leq N} \delta(t, t_i)$.

Note that if we let $B(t, r)$ denote the closed ball with center $t \in \mathfrak{T}$ and radius r , then the dispersion $d_N(T; \mathfrak{T})$ of T in $\mathfrak{T}$ may be described as the minimum of all radii $r \geq 0$ such that the balls $B(t_1, r), B(t_2, r), \dots, B(t_N, r)$ cover $\mathfrak{T}$.

Because the techniques of constructing low-dispersion sequences in Euclidean distance are not directly applicable in Hamming distance, Raphael et al. (2004) raised the following problem.

Uniform Projection Problem: Find a collection of (l, k) -templates $t_1, t_2, \dots, t_N$, such that each j -tuple $J = \{x_1, x_2, \dots, x_j\}$ of the l positions is covered by exactly λ templates.

This is equivalent to the problem of finding a j -(l, k, λ) design in combinatorial design theory (Beth et al., 1999), where j is the strength of the design. A solution to it gives a collection of templates with dispersion $d_N \leq 2(k - j)$, where N is the number of blocks in the j -(l, k, λ) design. Large values of j will yield templates with low dispersion, where blocks of the design are used as templates to replace the randomly chosen templates. Unfortunately, a j -(l, k, λ) design rarely exists for the values of l, k, j , and λ encountered in the motif finding problem, and furthermore, even a design exists for some particular values of l, k, j , and λ , constructing such a design explicitly is usually a difficult problem, especially for large j . An approximate solution to the uniform projection problem was given by Raphael et al. (2004) by setting $\lambda = 1$, and relaxing the condition to “each j -tuple $J = \{x_1, x_2, \dots, x_j\}$ of the l positions is covered by at least $\lambda = 1$ template.” This corresponds to a combinatorial structure called j -(l, k, λ) covering, where J is covered by exactly $\lambda_j (\geq \lambda)$ blocks. Furthermore, templates generated by a j -(l, k, λ) covering can hold the dispersion $d_N \leq 2(k - j)$, where N is the number of blocks in the j -(l, k, λ) covering.

Raphael et al. (2004) described a greedy approach to construct such coverings. As claimed by Raphael et al. (2004), the uniform projection algorithm performed better than the random projection algorithm, either by finding the motifs with fewer templates (20% fewer templates), or by finding the motifs in the cases where the random projection algorithm fails.

4. THE LOW-DISPERSION PROJECTION ALGORITHM

The success of this remains the same uniform projection algorithm gives us a hint of improving the projection algorithm. As a global optimization problem, the motif finding problem can be approached by quasi-random search methods that use suitable deterministic initial values instead of randomly chosen ones to reach the global optimization. Let f be an evaluation function of a motif finding algorithm, t be an initial value, and $f(t)$ be the evaluation result obtained by implementing the motif finding algorithm with the initial value t . Let $m(f) = \max_{t \in \mathfrak{T}} f(t)$ be the global maximum of f , where $\mathfrak{T}$ is the initial value space, and $m_N(f; T) = \max_{t \in T} f(t)$ be an estimate of $m(f)$, in which $T = \{t_1, t_2, \dots, t_N\} \subseteq \mathfrak{T}$. Also let $\omega(f; \delta') = \max_{t_i, t_j \in \mathfrak{T}, \delta(t_i, t_j) \leq \delta'} |f(t_i) - f(t_j)|$, where $\delta' \geq 0$ is a preset value. The following theorem can be found in Niederreiter (1992).

Theorem 1. If $\mathfrak{T}$ is a bounded metric space with Hamming distance δ , and f is continuous on $\mathfrak{T}$, then for any point set $T = \{t_1, t_2, \dots, t_N\}$ of N points in $\mathfrak{T}$ with dispersion $d_N(T; \mathfrak{T}) = \max_{t \in \mathfrak{T}} \min_{1 \leq n \leq N} \delta(t, t_n)$, we have

$$0 \leq m(f) - m_N(f; T) \leq \omega(f; d_N(T; \mathfrak{T})).$$

We take likelihood ratio as the evaluation function. Then Theorem 1 shows that suitable deterministic initial values for the projection algorithm are those templates with low dispersion. In Raphael et al.’s (2004) construction, the j -covering has a dispersion $d_N \leq 2(k - j)$. It is obvious that large values of j result in templates with low dispersion, but on the other hand, the number of templates increases correspondingly. For example, when dealing with the (15, 4)-motif finding problem, 172 templates were required by the random projection algorithm (Buhler and Tompa, 2002), but approximately 399.3 templates of size 7 with dispersion 4 were required to cover all 5-tuples to solve the problem (Raphael et al., 2004). So the first shortcoming of the uniform projection algorithm is that it needs too many templates to achieve “low-dispersion” property.

A related shortcoming of the uniform projection algorithm is that there is only a lower bound restriction to λ , that is $\lambda \geq 1$, in their construction of coverings by Raphael et al. (2004). If the indices λ_j differ too large for different j -tuple J , the j -tuple balance of the sampling will be broken, and consequently, the number of templates needed in the implementation of the uniform projection algorithm will increase. In our research, we further require an upper bound on λ in the following form. For the set of indices $\{\lambda_1, \lambda_2, \dots, \lambda_u\}$ of a covering, the value of $\frac{\min_{i \in \{1, 2, \dots, u\}} \{\lambda_i\}}{\max_{j \in \{1, 2, \dots, u\}} \{\lambda_j\}}$ should be as large as possible to guarantee the balance of the sampling.

The third shortcoming of the uniform projection algorithm is that using blocks of a covering as templates may result in overlapping projection results. For example, the following 2-(10, 7, 1) covering can be chosen to tackle the (10, 2)-motif finding problem by using the uniform projection algorithm:

$$t_1 = \{0, 2, 3, 4, 7, 8, 9\}, \quad t_2 = \{0, 1, 3, 4, 5, 6, 8\}, \quad t_3 = \{1, 2, 4, 5, 6, 7, 9\}.$$

We can see that $t_3 = t_2 + 1$. It is obvious that the projection outcomes resulted by templates t_2 and t_3 are overlapping. Clearly, performing t_3 in the projection algorithm is redundant, except that the projection of the last 10-mer onto t_3 should be added. The blocks (templates) that result in overlapping projection outcomes are said to be in the same track. In order to eliminate the overlapping projection outcomes, only one block should be chosen from each track to be the representative template.

The fourth shortcoming of the uniform projection algorithm is exposed when we analyze the template space. The template space can be viewed from the following way. Partitioning the template space by the last entry x of each template, that is, $k-1, k, \dots, l-1$, into $\mathcal{T}_{k-1}, \mathcal{T}_k, \dots, \mathcal{T}_{l-1}$. There are $\binom{x}{k-1}$ possible templates in $\mathcal{T}_x, x=k-1, k, \dots, l-1$. Large values of x yield large numbers of templates. That is, templates ended with large values of x should be sampled more frequently in the sampling. The covering, which treats each j -tuple as equal as possible, usually cannot reflect the distribution of the template space properly. Taking the (14, 4)-motif finding problem with template size 7 as an example, Figure 1 shows the average distribution of blocks of 20 randomly constructed 2-(14, 7, 1) coverings and the distribution of the template space of the (14, 7)-motif finding problem.

According to the above discussions, we know that templates should be selected from different tracks, with “low-dispersion” and “good balance” properties, and obey the distribution of the template space. The three requirements above guarantee the success of the projection algorithm in different ways. Furthermore, the projection algorithm always chooses an (l, k) -template t and projects every l -mer of background sequences onto t , where the l -mers are cyclically shifted from the beginning to the end of each background sequence. This further suggests the templates should be generated in the cyclic group of order l .

Let $\mathbb{Z}_l = \{0, 1, \dots, l-1\}$ be the cyclic group of order l , and $B = \{x_1, x_2, \dots, x_k\}$ be a k -subset of $\mathbb{Z}_l$, known as block. We denote $B+i = \{x+i \in \mathbb{Z}_l \mid x \in B\}$. The stabilizer of B is defined as $\mathbb{Z}_l(B) = \{i \in \mathbb{Z}_l \mid B+i = B\}$, and the orbit of B under $\mathbb{Z}_l$, or the $\mathbb{Z}_l$ -orbit of B , is defined as the set $B^{\mathbb{Z}_l}$ of distinct blocks $B^i = \{B+i \mid i \in \mathbb{Z}_l\}$. It is obvious that $\mathbb{Z}_l(B)$ is a subgroup of $\mathbb{Z}_l$ [4].

Let $\mathcal{B} = \{B_1, B_2, \dots, B_b\}$ be a set of blocks of size k of $\mathbb{Z}_l$. The multiset ΔB of differences from $B = \{x_1, x_2, \dots, x_k\} \in \mathcal{B}$ is defined as $\Delta B = \{x_s - x_t \mid 1 \leq s \neq t \leq k\}$, and the multiset $\Delta \mathcal{B}$ of differences from $\mathcal{B}$ is defined as $\Delta \mathcal{B} = \bigcup_{B \in \mathcal{B}} \frac{\Delta B}{|\mathbb{Z}_l(B)|}$, where the multiplicities are also counted.

FIG. 1. The distributions of the template space and blocks of a covering.

For example, we consider a set $\mathcal{B} = \{B_1, B_2, B_3\}$ of blocks of size 3 of $\mathbb{Z}_{15}$, where $B_1 = \{0, 1, 4\}$, $B_2 = \{0, 7, 13\}$, $B_3 = \{0, 5, 10\}$. In this case, the $\mathbb{Z}_{15}$ -orbits are

$$\begin{aligned} B_1^{\mathbb{Z}_{15}} &= \{\{0, 1, 4\}, \{1, 2, 5\}, \{2, 3, 6\}, \{3, 4, 7\}, \{4, 5, 8\}, \{5, 6, 9\}, \{6, 7, 10\}, \{7, 8, 11\}, \\ &\quad \{8, 9, 12\}, \{9, 10, 13\}, \{10, 11, 14\}, \{11, 12, 0\}, \{12, 13, 1\}, \{13, 14, 2\}, \{14, 0, 3\}\}, \\ B_2^{\mathbb{Z}_{15}} &= \{\{0, 7, 13\}, \{1, 8, 14\}, \{2, 9, 0\}, \{3, 10, 1\}, \{4, 11, 2\}, \{5, 12, 3\}, \{6, 13, 4\}, \{7, 14, 5\}, \\ &\quad \{8, 0, 6\}, \{9, 1, 7\}, \{10, 2, 8\}, \{11, 3, 9\}, \{12, 4, 10\}, \{13, 5, 11\}, \{14, 6, 12\}\}, \\ B_3^{\mathbb{Z}_{15}} &= \{\{0, 5, 10\}, \{1, 6, 11\}, \{2, 7, 12\}, \{3, 8, 13\}, \{4, 9, 14\}\} \end{aligned}$$

the differences are

$$\Delta B_1 = \{1, 3, 4, 11, 12, 14\}, \Delta B_2 = \{2, 6, 7, 8, 9, 13\}, \Delta B_3 = \{5, 5, 5, 10, 10, 10\},$$

the stabilizers are

$$\mathbb{Z}_{15}(B_1) = \{0\}, \mathbb{Z}_{15}(B_2) = \{0\}, \mathbb{Z}_{15}(B_3) = \{0, 5, 10\},$$

and finally

$$\begin{aligned} \frac{\Delta B_1}{|\mathbb{Z}_{15}(B_1)|} &= \{1, 3, 4, 11, 12, 14\}, \frac{\Delta B_2}{|\mathbb{Z}_{15}(B_2)|} = \{2, 6, 7, 8, 9, 13\}, \frac{\Delta B_3}{|\mathbb{Z}_{15}(B_3)|} = \{5, 10\}, \\ \Delta \mathcal{B} &= \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14\}. \end{aligned}$$

It is easily seen that the 35 blocks in the $\mathbb{Z}_{15}$ -orbits of B_1, B_2 , and B_3 form a 2-(15, 3, 1) design. It should be noted that although $\{0, 1, 4\}$, $\{11, 12, 0\}$, and $\{14, 0, 3\}$ are in the same orbit $B_1^{\mathbb{Z}_{15}}$, they are not in the same track. The blocks in the same track with $\{0, 1, 4\}$ are $\{1, 2, 5\}, \{2, 3, 6\}, \{3, 4, 7\}, \{4, 5, 8\}, \{5, 6, 9\}, \{6, 7, 10\}, \{7, 8, 11\}, \{8, 9, 12\}, \{9, 10, 13\}$, and $\{10, 11, 14\}$. The blocks in the same track with $\{11, 12, 0\}$ are $\{12, 13, 1\}$ and $\{13, 14, 2\}$. There is no other block in the same track with $\{14, 0, 3\}$.

Almost Difference Family: Let $1 \leq \lambda_1 < \lambda_2 < \dots < \lambda_u$ be u distinct integers. Let $X_1, X_2, \dots, X_u$ be a partition of $\mathbb{Z}_l \setminus \{0\}$, and $\mathcal{B}$ a set of blocks of size k of $\mathbb{Z}_l$. If every element of X_i , $1 \leq i \leq u$, appears in $\Delta \mathcal{B}$ exactly λ_i times, we say that $\mathcal{B}$ is an almost difference family, briefly denoted by $(l, k, \{\lambda_1, \lambda_2, \dots, \lambda_u\})$ -ADF.

Developed Almost Difference Family: The set $\mathcal{B}^*$ formed by blocks of size k containing the identity element $0 \in \mathbb{Z}_l$ in all the $\mathbb{Z}_l$ -orbits of an $(l, k, \{\lambda_1, \lambda_2, \dots, \lambda_u\})$ -ADF is called a developed almost difference family, briefly denoted by $(l, k, \{\lambda_1, \lambda_2, \dots, \lambda_u\})$ -DADF.

Note that the blocks in an ADF are the representations of all $\mathbb{Z}_l$ -orbits, while the blocks in a DADF are the representations of all tracks.

It is easily seen (Beth et al., 1999) that all the blocks in the $\mathbb{Z}_l$ -orbits of an $(l, k, \{\lambda_1, \lambda_2, \dots, \lambda_u\})$ -ADF form a 2-(l, k, λ_1) covering, the 2-(l, k, λ_1) covering with larger $\frac{\lambda_1}{\lambda_u}$ has fewer blocks, and the 2-(l, k, λ_1) covering with $\frac{\lambda_1}{\lambda_u} = 1$ is a 2-(l, k, λ_1) design. To improve the uniform projection algorithm, we should use the blocks of those 2-(l, k, λ_1) coverings with large $\frac{\lambda_1}{\lambda_u}$ as templates, which can also assure their dispersions $d_N \leq 2(k-2)$. In the remainder of this article, we always require $\frac{\lambda_1}{\lambda_u} \geq 0.85$. We also note that some blocks of all $\mathbb{Z}_l$ -orbits of an ADF may be in the same tracks, so we should choose the blocks of the corresponding $(l, k, \{\lambda_1, \lambda_2, \dots, \lambda_u\})$ -DADF as templates to promote efficiency by a factor of $\frac{k}{j}$. Any value of the record in the hash table obtained by applying a template $t = \{t_1, t_2, \dots, t_k\}$ should be multiplied by a factor $l - t_k$, where t_k represents the largest entry of t , since $l - t_k - 1$ templates in the same track with t are omitted. In this way, we say that the blocks in a DADF can cover a j -tuple if and only if the blocks in all $\mathbb{Z}_l$ -orbits of the corresponding ADF can do it.

It is an additional benefit if the $(l, k, \{\lambda_1, \lambda_2, \dots, \lambda_u\})$ -ADF we constructed has the property that the blocks in its $\mathbb{Z}_l$ -orbits in fact form a j -covering with $j > 2$, so that the dispersion of the blocks can be assured to be $d_N \leq 2(k-j) < 2(k-2)$. In our construction, we first construct an $(l, k, \{\lambda_1, \lambda_2, \dots, \lambda_u\})$ -ADF with $\frac{\lambda_1}{\lambda_u} \geq 0.85$, then check if the blocks in its $\mathbb{Z}_l$ -orbits form a j -covering with $j > 2$. If the blocks cannot form a j -covering with $j > 2$, we discard it and construct a new one. The process continues until the blocks in the $\mathbb{Z}_l$ -orbits of an $(l, k, \{\lambda_1, \lambda_2, \dots, \lambda_u\})$ -ADF can form a j -covering with $j > 2$.

Compared with the j -coverings constructed by Raphael et al. (2004) in the uniform projection algorithm, fewer blocks in DADFs are required to cover all j -tuples. For fixed values of j and l , Table 1 gives a detailed

TABLE 1. TEMPLATE NUMBERS REQUIRED TO COVER j -TUPLES BY DADF/RAPHAEL ET AL.'S j -COVERING

	$l = 14$	$l = 15$	$l = 16$	$l = 17$	$l = 18$	$l = 19$
$j = 3$	14/24.7	14/31.7	21/39	21/48.3	21/58.9	21/71.4
$j = 4$	28/74.8	28/105.6	63/143.9	63/192	84/249.9	112/321.1
$j = 5$	119/258.8	182/399.3	322/591	420/845.9	588/1185.8	826/1624

DADF, developed almost difference family.

comparison between the numbers of blocks required to cover j -tuples by DADFs and Raphael et al.'s j -coverings respectively. The data for DADFs are from the detailed constructions for $(l, 7, 1)$ -DADFs. The data for j -coverings are from Raphael et al. (2004).

Therefore, for each listed value of l , the numbers of blocks needed to cover j -tuples by DADFs are less than one-third to one-half of those by Raphael et al.'s j -coverings.

In this way, the $(l, k, \{\lambda_1, \lambda_2, \dots, \lambda_u\})$ -DADF we constructed can generate a low-dispersion sequence $d_N \leq 2(k - j)$ with a small number of blocks, where j represents the strength of the corresponding covering. At this point, choosing the blocks of the $(l, k, \{\lambda_1, \lambda_2, \dots, \lambda_u\})$ -DADF as templates not only can promote the efficiency of the uniform projection algorithm by a factor of $\frac{k}{j}$, but also can make the algorithm reaching the global maximum by Theorem 1. In the following, we briefly denote such an $(l, k, \{\lambda_1, \lambda_2, \dots, \lambda_u\})$ -DADF as j -(l, k)-DADF.

Furthermore, our experiment shows that the distribution of blocks of a j -(l, k)-DADF basically agrees with the distribution of the template space. Figure 2 shows the average distribution of blocks of 20 randomly constructed 2-(14, 7)-DADFs, the template space of (14, 7)-motif finding problem, and the average distribution of the blocks of 20 randomly constructed 2-(14, 7, 1)-coverings. It can be seen that the distribution of blocks of DADFs and the distribution of the template space are nearly the same.

The projection algorithm using blocks of DADF as templates is called the low-dispersion projection algorithm in this article. We believe that using the blocks of DADFs as templates can provide a better sampling on the template space, promote the efficiency of the projection algorithm, and eventually result in the global maximum of the motif finding problem.

5. RESULTS

Buhler (2001) described the random projection algorithm in his PhD thesis, and designed a corresponding C++ implementation called Projection Genomics Toolkit for the random projection algorithm. Projection Genomics Toolkit can be downloaded online. According to the illustration of Projection Genomics Toolkit, the implementation performed well when l is equal to 15~20 and d is equal to 0.1~0.25 l .

FIG. 2. The distribution of blocks of DADFs, the distribution of the template space, and the distribution of blocks of coverings.

TABLE 2. EXPERIMENTAL RESULTS OF THE RANDOM PROJECTION ALGORITHM
VS. THE LOW-DISPERSION PROJECTION ALGORITHM

l	d	SC of RPA	SC of LDPA	Template nb. of RPA	Template nb. of LDPA
15	3	0.98	0.98	47	24
16	4	0.95	0.95	462	128
17	4	0.97	0.97	155	40
18	5	0.89	0.95	1205	256
19	5	0.93	0.95	413	64

In order to compare the low-dispersion projection algorithm with the random projection algorithm, we modified Buhler's C++ implementation by changing the random template generation routine to our DADF construction scheme.

It should be noted that in Buhler's implementation (2001), the hash table threshold is $s = 4$, the template size is $k = 8$, and the number of iterations is

$$m = \left\lceil \frac{\log(1-q)}{\log B_{h,p}(s)} \right\rceil.$$

In our implementation, we continue to use the parameters $s = 4$ and $k = 8$. We first construct a DADF with block size 8, then use the blocks of the DADF as templates. The construction of DADF is executed according to the following principle: on the precondition of $j \geq 4$, we require that the number of blocks b be as small as possible, so that each j' -tuple with $j' \leq j$ is covered by the blocks as evenly as possible. Although large j decreases the dispersion of blocks, the number of blocks b will also increase correspondingly. In our implementation, we take $j = 4$ or $j = 5$.

Table 2 shows a comparison between the random projection algorithm (RPA) and the low-dispersion projection algorithm (LDPA) in dealing with the (l, d) -motif finding problem on $h = 20$ DNA sequences of length $n = 600$. For each (l, d) -motif finding problem, average performance coefficients of the low-dispersion projection algorithm over 100 random experiments are reported. Setting n_f to be the number of instances that an algorithm can not recover the planted consensus, the success rate is defined as $SC = 1 - \frac{n_f}{100}$.

For those instances that the random projection algorithm can solve the motif finding problem successfully, the low-dispersion projection algorithm can also achieve success. When both the random projection algorithm and the low-dispersion projection algorithm failed, sometimes the low-dispersion projection algorithm can recover motifs that are closer to the planted motifs in position. When solving the (19, 5)-, (18, 5)-motif finding problems, the low-dispersion projection algorithm achieves a higher success rate than the random projection algorithm. Meanwhile, compared with the random projection algorithm, at most half of the templates are needed to solve the same motif finding problem by the low-dispersion projection algorithm. The costs in computational time and memory for the construction of the corresponding DADFs can be ignored, and the reduction in running times using the low-dispersion projection versus random projection approximately equals the reduction in the number of templates (15~50%), because the most time-consuming step of both algorithms is the EM refinement strategy.

6. CONCLUDING REMARKS

In this article, we improved the initial value-choosing strategy of the projection algorithm. We first proved that, as a global optimization problem, the motif finding problem can be optimized by using initial values that form low-dispersion sequences. Then we introduced the notion of developed almost difference families (DADFs), which can generate low-dispersion sequences. Finally, we improved the random projection algorithm by using the blocks of DADFs as templates. We showed that using the blocks of DADFs can promote the efficiency of the projection algorithm and make the algorithm reach the global optimization. Experiments also showed that, compared with the random projection algorithm, the low-dispersion projection algorithm can solve the motif finding problem with less templates (15~50%) without decreasing the success rate.

The low-dispersion sequences generated by our construction can be applied to other problems where the projection algorithm is proved to be useful, such as nearest neighbor search (Gionis et al., 1999), dimension reduction (Wang et al., 2008), and cluster analysis (Fern and Brodley, 2003).

ACKNOWLEDGMENTS

This work was supported in part by the JSPS Grant-in-Aid for Scientific Research (C) under Grant No. 24540111, as well as the National Science Foundation of China under Grant No. 11301098, Guangxi Natural Science Foundation under Grant No. 2013GXNSFCA019001, Foundation of Guangxi Education Department under Grant No. 2013YB039, and the Scientific Research Foundation for the Returned Overseas Chinese Scholars, State Education Ministry.

AUTHOR DISCLOSURE STATEMENT

There is no competing financial interest among the authors.

REFERENCES

- Bailey, T.L., and Elkan, C. 1995. Unsupervised learning of multiple motifs in biopolymers using expectation maximization. *Machine Learning* 21, 51–80.
- Beth, T., Jungnickel, D., and Lenz, H. 1999. *Design Theory*. Cambridge University Press, Cambridge, United Kingdom.
- Buhler, J. 2001. Search algorithms for biosequences using random projection [PhD Thesis]. University of Washington.
- Buhler, J., and Tompa, M. 2002. Finding motifs using random projections. *J. Comput. Biol.* 9, 225–242.
- Das, M.K., and Dai, H.K. 2007. A survey of DNA motif finding algorithms. *BMC Bioinformatics* 8, S21.
- Fern, X.L., and Brodley, C.E. 2003. Random projection for high dimensional data clustering: a cluster ensemble approach. *Proc. 20th Int. Conf. Machine Learning*, 186–193.
- Galas, D.J., and Schmitz, A. 1978. DNase footprinting: a simple method for the detection of protein-DNA binding specificity. *Nucleic Acids Res.* 5, 3157–3170.
- Garner, M.M., and Revzin, A. 1981. A gel electrophoresis method for quantifying the binding of protein to specific DNA regions: application to components of the Escherichia coli lactose operon regulatory system. *Nucleic Acids Res.* 9, 3047–3060.
- Gionis, A., Indyk, P., and Motwani, R. 1999. Similarity search in high dimensions via hashing. *Proc. 25th Int. Conf. Very Large Data Bases*, 518–529.
- Hertz, G.Z., and Stormo, G.D. 1999. Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics* 15, 563–577.
- Lawrence, C.E., Altschul, S.F., Boguski, M.S., et al. 1993. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science* 262, 208–214.
- Lawrence, C.E., and Reilly, A.A. 1990. An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins* 7, 41–51.
- Li, N., and Tompa, M. 2006. Analysis of computational approaches for motif discovery. *Algorithms Mol. Biol.* 1, 8.
- Niederreiter, H. 1992. Random number generation and quasi-Monte Carlo methods. *CBMS-NSF Regional Conf. Series in Applied Math.* 63.
- Pevzner, P.A., and Sze, S.H. 2000. Combinatorial approaches to finding subtle signals in DNA sequences. *Proc. Int. Conf. Intell. Syst. Mol. Biol.*, 269–278.
- Raghavarao, D. 1988. *Constructions and Combinatorial Problems in Design of Experiments*. Dover, New York.
- Raphael, B., Liu, L.T., and Varghese, G. 2004. A uniform projection method for motif discovery in DNA sequences. *IEEE/ACM Trans. Comput. Biol. Bioinform.* 1, 91–94.
- Wang, H.S., Ni, L.Q., and Tsai, C.L. 2008. Improving dimension reduction via contour-projection. *Statistica Sinica* 18, 299–311.

Address correspondence to:

Xun Wang

Tennodai 1-1-1

Graduate School of Systems and Information Engineering

University of Tsukuba

Tsukuba 305-8577

Japan

E-mail: wangxun0830@hotmail.com