Designing universal oligonucleotides for DNA/nanoparticle conjugates

Gabor Ivan[1,2], Vince Grolmusz[1,2]

[1] Eotvos Lorand University
Pazmany Peter stny. 1/C., H-1117 Budapest, Hungary; hugeaux@cs.elte.hu
[2] Uratim Ltd.
InfoPark Building D, Gabor Denes u. 2. H-1117 Budapest, Hungary; grolmusz@cs.elte.hu

ABSTRACT

As DNA sequences originating from different species are highly specific, it turns out to be a challenge to design DNA/nanoparticle conjugates that are "universal" in the sense that they are able to hybridize to each member of a set of nucleotide sequences. Possible applications of such conjugates include the detection of different nucleotide sequences using only a few types of DNA/nanoparticle conjugates.

In this paper we present a reasonably fast computational method suitable to design oligonucleotide probes that *cover* a given input set of nucleotide sequences. The proposed algorithm reduces the (otherwise exponential-sized) search space by discarding inadequate oligonucleotides as soon as possible. We investigate the results of the method using a wide range of parameter settings and also propose ideas for further improvement.

Keywords: DNA/nanoparticle conjugates, nanoparticle design, minimal oligonucleotide cover set, APRIORI algorithm, set covering problem

1 INTRODUCTION

An important application of nano-size particles is the detection of certain biomolecules. In these applications, nanoparticles are constructed so that they bind to specific biomarkers; the bound nanoparticles make optical recognition of the biomarkers possible [3]. One specific application of nanoparticles is sensitive, specific DNA detection: it has the potential to provide a lot of information from a small sample volume at low cost, thus it is of great demand for various biological and biomedical studies (e.g. gene profiling, clinical diagnostics etc.) [4], [1]. Attention has also been dedicated to the precise assembly of DNA/nanoparticle conjugates, i.e. the development of methods that result in good quality conjugates [2].

1.1 Notation and basic definitions

A *DNA/nanoparticle conjugate* is a molecule consisting of a nano-sized particle coupled to an oligonucleotide. DNA/nanoparticle conjugates might be used for marking DNA molecules, as the oligonucleotide-part of the conjugate may hybridize to a specific part of the DNA molecule, and then the non-mobile nanoparticle-part can be detected. We currently deal with the design of the oligonucleotide part when the aim is to manufacture nanoparticles that bind to the most possible number of DNA sequences. It is assumed that a few mismatches (e.g. one in every ten nucleotides) can be tolerated and do not interfere with hybridization.

We are going to denote sets with capital letters (e.g. $\mathcal{N}, \mathcal{T}, \mathcal{O}$) and the cardinality of a set by $|\mathcal{N}|$. If n denotes a sequence, we are going to denote its length by $|n|$.

1.2 Problem description

The problem we are about to manage is as follows. We are given a set $\mathcal{N}$ of several nucleotide sequences. Our task is to find a set of oligonucleotides $\mathcal{O}$ of the smallest possible cardinality such that the oligonucleotides $o \in \mathcal{O}$ together *cover* all the sequences contained in $\mathcal{N}$.

An oligonucleotide o *covers* a nucleotide sequence $n \in \mathcal{N}$ if – at some position $p \in \{1...|n| - |o| + 1\}$ – it occurs in n with at most e mismatches (where e is a small, pre-defined integer, $0 \leq e \leq 4$). We are going to call such an oligonucleotide a *cover-sequence* for n.

An oligonucleotide set $\mathcal{O}$ *covers* a nucleotide set $\mathcal{N}$ if every nucleotide sequence $n \in \mathcal{N}$ is *covered* by at least one oligonucleotide $o \in \mathcal{O}$. We are going to call such an oligonucleotide set a *cover-set* for $\mathcal{N}$.

It is trivial that the above problem can always be solved by choosing $|\mathcal{O}| = |\mathcal{N}|$ by picking a substring of each nucleotide $n \in \mathcal{N}$ and adding this arbitrarily chosen substring to $\mathcal{O}$. Therefore, we are interested in solutions (oligonucleotide sets) with cardinality less than the cardinality of $\mathcal{N}$.

It is noteworthy that – for practical application purposes – we are going to require an oligonucleotide $o \in \mathcal{O}$ to be of length approx. 20-50 nucleotides. This means that in the nontrivial case, testing every possible oligonucleotide with a length of up to e.g. 50 is infeasible, as the size of the search space is an exponential function of oligonucleotide length. [1] Moreover, we are not after one single oligonucleotide, but the smallest (or nearly smallest) possible set of them, which further complicates the problem. Previous papers dealing with covers of a sequence of characters [5], [6] use a different

[1] There are $|\{A, C, G, T\}|^l = 4^l$ possibilities for oligonucleotides of length l.

definition of the *covering* of a string, based on different motivations.

2 METHODS

We are going to divide the work leading to the set $\mathcal{O}$ into two phases.

1. In the first phase we are going to search for the set $\mathcal{T}$ of all the oligonucleotides that *cover* at least a pre-defined number of nucleotide sequences of $\mathcal{N}$ with at most e errors. The minimum required number of *covered* sequences $n \in \mathcal{N}$ is furthermore denoted by m. Finding the set $\mathcal{T}$ is the main focus of this work.

2. In the second phase, we try to choose the set $\mathcal{O}$ from the elements of $\mathcal{T}$ so that $\mathcal{O}$ is a *cover-set* of minimal cardinality for $\mathcal{N}$.

The solution to our problem is not too hard to compute (e.g. using generalized suffix trees) if the sequences in $\mathcal{N}$ share a sufficiently long common substring; in this case, this substring may be chosen as the one and only element of $\mathcal{O}$, (without even trying to determine $\mathcal{T}$). Note that searching common substrings for all the elements of $\mathcal{N}$ corresponds to setting $e = 0$ and $m = |\mathcal{N}|$. However, with real-world DNA sequences in $\mathcal{N}$, the above situation proves to be rarely the case: even DNA sequences from the same species (but different strains) may be different enough that they do not have an *exact* common substring of sufficient length.

2.1 The main idea and relation to the APRIORI algorithm

There is a method called the *APRIORI algorithm*[7] that is well-known for researchers involved in the field of data mining. The APRIORI algorithm finds frequently occurring sets of items in a set of transactions (where each transaction itself is a set of items): an itemset is defined "frequent" if all of its items occur together in at least q transactions. The main idea of the APRIORI algorithm is *"Every subset of a frequent itemset is frequent."* In other words this means that we cannot make an infrequent itemset frequent by adding more items to it.

Analogously, the hereby proposed method relies on a very similar observation: *Every substring of a cover-sequence is a cover-sequence.* In other words, we cannot reduce the number of errors (mismatches) in any gapless alignment of two or more sequences by adding more nucleotides to an arbitrary sequence, as the newly added nucleotides might or might not induce new errors, but do not have an influence on existing ones. This also implies the following: if it turns out that some oligonucleotide t does not cover a sequence $n \in \mathcal{N}$, nor will any t_1 oligonucleotide that contains t as a substring.

Based on this observation, our algorithm works as follows:

1. Initialization: Begin with the list of all possible 4-sized oligonucleotides (there are $4^4 = 256$ of them) furthermore referred to as *candidates*. For each of these candidates we maintain a *sequence-list* of covered sequences $n \in \mathcal{N}$, and a *position-list*: for the covered sequences, the first position in each sequence that results in $\leq e$ mismatches when aligning the candidate at that position. At the beginning, the *sequence-list* contains (for every candidate) all the sequences in $\mathcal{N}$, and every candidate's *position-list* is initialized to 1 for all sequences in in $\mathcal{N}$.

2. Check each candidate: discard sequences from its *sequence-list* of covered sequences if necessary. Also update *position-lists* for covered sequences. If a candidate turns out not to cover at least m sequences, discard the candidate itself. It is worthwile to note that this step can be vastly speeded up by utilizing SIMD instruction sets (e.g. SSE2) of modern processors.

3. Prepare new candidates one nucleotide longer based on the ones currently tested and test these new candidates at the previous step. This last step requires more thorough inspection.

2.2 Preparing new candidates

In addition to the algorithm details added above, we begin with a lexicographically ordered list of candidates. We heavily rely on this property and maintain it throughout the new candidate preparation part of the algorithm. Let us assume that we have just finished testing candidates of length l. We are going to use two candidates of length l to produce one candidate of length $l+1$. A candidate of length $l+1$ is only worth considering if both its l-length prefix and its l-length suffix are a *cover-sequence* for at least m sequences in $\mathcal{N}$, and the intersection of *sequence-lists* of these candidates also contains at least m sequences. The l-length suffix s and the l-length prefix p of an oligonucleotide of length $l+1$ may only differ in their first and last nucleotides. (E.g. `AGGTCAG`: `AGGTCA`, `GGTCAG`.) Based on the above observations, we proceed as follows. We concatenate all four nucleotides $\{A, C, G, T\}$ to the end of each successfully tested candidate. Then we search for the l-length suffix of each such new candidate in the list of the currently tested candidates of length l. (For example, concatenating an `'G'` to the candidate `AGGTCA` leads to searching for the already-tested candidate `GGTCAG`.) As the list of candidates is lexicographically ordered, binary search may be applied. If the l-length suffix s is not found, it means that s was not a *cover-sequence* for at least m sequences in $\mathcal{N}$. Therefore, no sequence containing s can cover m

sequences from $\mathcal{N}$: the new candidate of length $l+1$ is discarded.

Otherwise, it remains for us to discuss how to calculate the sequence-list and position-list for the new candidate.

- The *sequence-list* is the intersection of the sequence lists of p and s.
- The *position-list* (for sequences in the above intersection) contains the higher value of the two position-lists. [2]

It is easy to see that the lexicographical order of the candidates can easily be preserved. To see this, one has to consider the candidates' equal length and that it is possible to concatenate the last character so that the ordering remains preserved.

3 RESULTS AND DISCUSSION

3.1 Results of phase 1 (determining $\mathcal{T}$)

Investigation of results based on the above algorithm was carried out using 35 strains of HPV (*Human Papillomavirus*) genomes found in the *RefSeq* sequence database.

Figure 1 shows the number of oligonucleotides that cover at least $m = 2$ genomes as a function of oligonucleotide length for different allowed error rates ($e = \{0, 1, 2\}$). For small values of oligonucleotide length l, all the 4^l possible oligonucleotides *cover* the minimally required $m = 2$ genomes; for bigger values of l, more and more oligonucleotides are discarded. Note that the y axis is scaled logarithmically, Figure 1 thus indicates that the number of candidates at first rises exponentially, and then falls exponentially.

Based solely on Figure 1, one may conclude that it is easy to gain a significant number of oligonucleotides that *cover* at least m sequences in $\mathcal{N}$. This is true, but usually quite a large number of oligonucleotides belong to the same subset of sequences $\mathcal{S} \subseteq \mathcal{N}$. If, for example, two sequences are identical in $\mathcal{N}$ and $m = 2$, all the substrings of these sequences will be outputted, together with all possible strings that are at most e Hamming distance from these substrings. The situation is very similar if two or more sequences share a long common substring.

Because of the above observation, it seems to be advisable to count the number of oligonucleotides that are *cover-sequences* for sequences $n \in \mathcal{N}$. The three figures in Figure 2 summarize the dependence of the number of *cover-sequences* from the parameters e, m for each genome; oligonucleotide length is fixed at $l = 20$. As the y axis is again scaled logarithmically, we may conclude that some sequences prove to be much "harder" to cover

[2] If the selected "higher value" is in the position-list of s, we also have to subtract 1 from it.

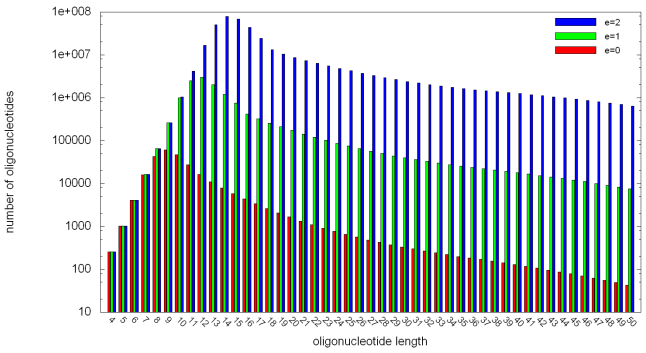

Figure 1: Number of oligonucleotides found as a function of length (x axis) and error rate e; $m = 2$.

than others. These are highly different from any other sequence in the input dataset; see for example genome No. 18 in Figure 2.

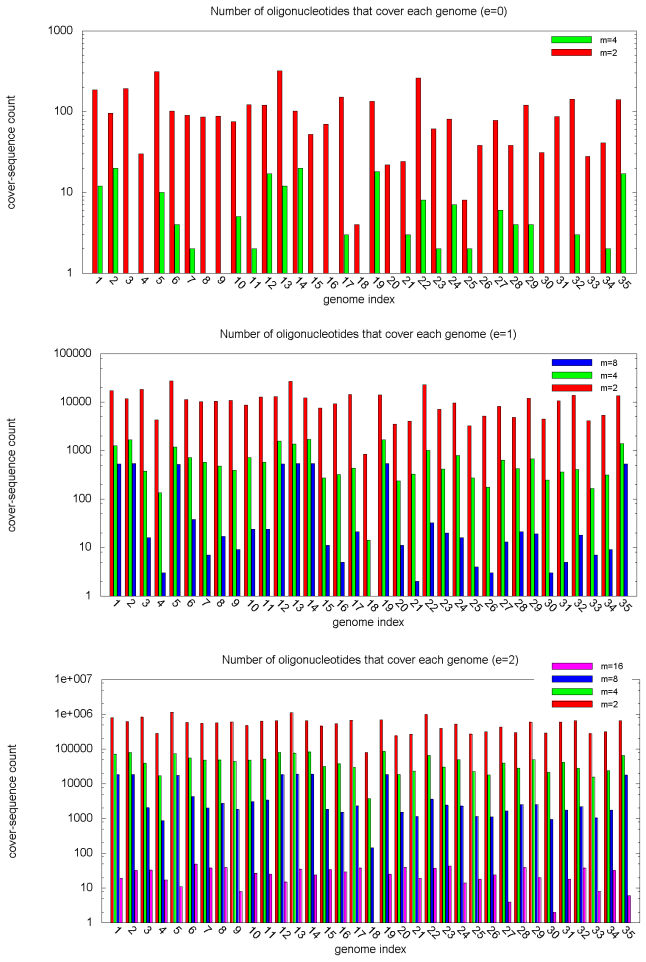

Figure 2: Dependence of the number of *cover-sequences* from the parameters e, m for each of the 35 HPV-genomes. Length of the *cover-sequences* is fixed at 20.

3.2 Results of phase 2 (determining $\mathcal{O}$)

After the first phase we are left with a set $\mathcal{T}$ consisting of several oligonucleotides that *cover* some sequences of $\mathcal{N}$ (with at most e errors).

The remaining task is choosing a set $\mathcal{O} \subseteq \mathcal{N}$ of minimal cardinality such that $\mathcal{O}$ is a *cover-set* for $\mathcal{N}$. As this problem is analogous to the *set cover problem* which is NP-complete, we hereby propose two simple greedy approaches to choose a *cover-set* of hopefully small cardinality.

1. We choose one oligonucleotide t from those that *cover* the most sequences in $\mathcal{N}$. The sequences t *covers* are removed from $\mathcal{N}$, t is outputted and removed from $\mathcal{T}$. This is repeated until the set $\mathcal{N}$ is empty.

2. Very similar to the procedure above; we choose an oligonucleotide t from those that *covers* the most sequences in $\mathcal{N}$ while also covering the sequence $s \in \mathcal{N}$ that has the least *cover-sequences* belonging to s in $\mathcal{T}$.

In both cases above, the outputted $t_1, t_2, ..., t_{|\mathcal{O}|}$ oligonucleotides constitute the final oligonucleotide set $\mathcal{O}$. Composition of the final oligonucleotide sets using different parameter settings are depicted in Figure 3. If $e = 2$ and $m \leq 8$, the whole $\mathcal{N}$ can be *covered* with only 3 different oligonucleotides of length 20 with at most $e = 2$ errors. Usually the cardinality of $\mathcal{O}$ gained by the above greedy approaches is so small that even brute-force algorithms that try all possible aligonucleotide pairs, triplets etc. (up to $(|\mathcal{O}| - 1)$-tuples) from $\mathcal{T}$ may come into consideration, if one would like to find the exact optimum instead of just an approximation.

4 CONCLUSION

We obtained a method with reasonable computational resource requirement that finds nearly optimal *covers* (oligonucleotides) of the input nucleotide set $\mathcal{N}$. Further development possibilities include favoring oligonucleotides that cover the same DNA sequence $n \in \mathcal{N}$ at several different positions, or investigating the effect of using restriction endonucleases on the set $\mathcal{N}$ to obtain shorter sequences to cover.

Being able to optimize the oligonucleotide sequences attached to a nanoparticle, we believe our proposed method will prove to be useful in several areas of biotechnology where DNA/nanoparticle conjugates are involved.

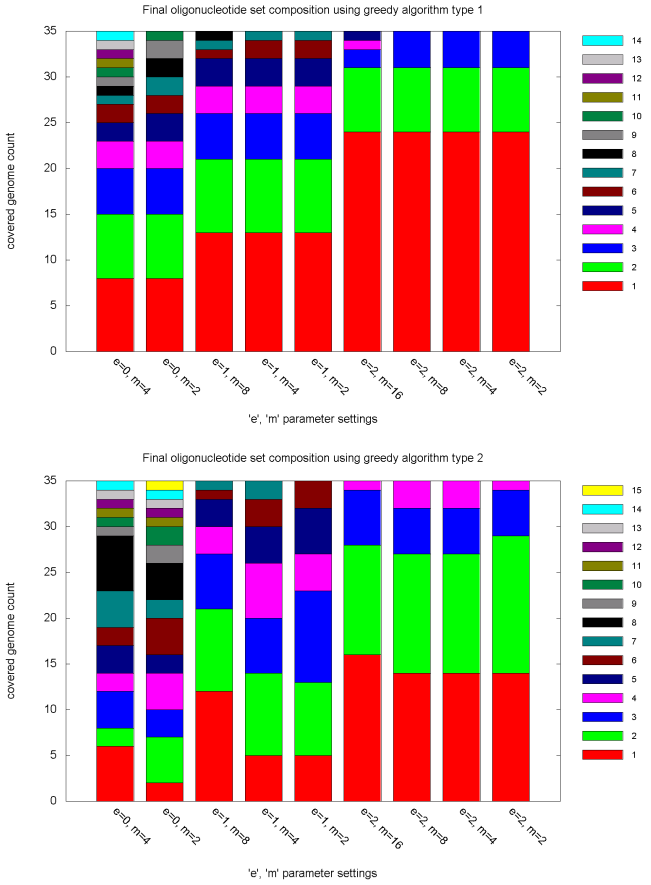

Figure 3: Constitution of *cover-sets* obtained from $\mathcal{T}$ using greedy algorithms, with different parameter settings of e and m. *Cover-sequence* length is fixed at 20.

REFERENCES

[1] Ken Adachi, Naohiro Noda, Makoto Nakashige, Satoshi Tsuneda, Takahiro Kanagawa, *Affinity capillary electrophoresis with a DNA-nanoparticle conjugate as a new tool for genotyping*, Journal of Chromatography A, 1109 (2006), pp. 127-131.

[2] Christopher J. Ackerson, Michael T. Sykes, Roger D. Kornberg, *Defined DNA/nanoparticle conjugates*, PNAS Vol. 102, No. 38, pp. 13383-13385 (September 20, 2005).

[3] Sarah H Radwan and Hassan M E Azzazy, *Gold nanoparticles for molecular diagnostics*, Expert Rev Mol Diagn, 9(5):511–524, Jul 2009.

[4] Taejoon Kang, Seung Min Yoo, Ilsun Yoon, Sang Yup Lee, Bongsoo Kim, *Patterned Multiplex Pathogen DNA Detection by Au Particle-on-Wire SERS Sensor*, to appear in Nano Letters, doi: 10.1021/nl1000086, March 2010.

[5] Dennis Moore, W. F. Smyth, *An Optimal Algorithm To Compute All The Covers Of A String*, Information Processing Letters Vol. 50 (1994), pp. 101-103.

[6] R. Cole, C.S. Iliopoulos, M. Mohamed, W.F. Smyth, L. Yang, *Computing the minimum k-cover of a string*, Proceedings of the 2003 Prague Stringology Conference, Prague, 2003, pp. 51-64.

[7] R. Rakesh Agrawal, Tomasz Imielinski, Arun Swami, *Mining association rules between sets of items in large databases*, SIGMOD June 1993, Vol. 22., No. 2., pp. 207-216.