Improved Tag Set Design and Multiplexing Algorithms for Universal Arrays*

Ion I. Măndoiu, Claudia Prăjescu, and Dragoş Trincă

CSE Department, University of Connecticut
371 Fairfield Rd., Unit 2155, Storrs, CT 06269-2155
{ion.mandoiu,claudia.prajescu,dragos.trinca}@uconn.edu

Abstract. In this paper we address two optimization problems arising in the design of genomic assays based on universal tag arrays. First, we address the universal array tag set design problem. For this problem, we extend previous formulations to incorporate antitag-to-antitag hybridization constraints in addition to constraints on antitag-to-tag hybridization specificity, establish a constructive upper bound on the maximum number of tags satisfying the extended constraints, and propose a simple alphabetic tree search tag selection algorithm. Second, we give methods for improving the multiplexing rate in large-scale genomic assays by combining primer selection with tag assignment. Experimental results on simulated data show that this integrated optimization leads to reductions of up to 50% in the number of required arrays.

1 Introduction

High throughput genomic technologies have revolutionized biomedical sciences, and progress in this area continues at an accelerated pace in response to the increasingly varied needs of biomedical research. Among emerging technologies, one of the most promising is the use of *universal tag arrays* [4, 7, 9], which provide unprecedented assay customization flexibility while maintaining a high degree of multiplexing and low unit cost.

A universal tag array consists of a set of DNA *tags*, designed such that each tag hybridizes strongly to its own *antitag* (Watson-Crick complement), but to no other antitag [2]. Genomic assays based on universal arrays involve multiple hybridization steps. A typical assay [3, 5], used for Single Nucleotide Polymorphism (SNP) genotyping, works as follows. (1) A set of *reporter oligonucleotide probes* is synthesized by ligating antitags to the $5'$ end of primers complementing the genomic sequence immediately preceding the SNP location in $3'$-$5'$ order on either the forward or reverse strands. (2) Reporter probes are hybridized in solution with the genomic DNA under study. (3) Hybridization of the primer part ($3'$ end) of a reporter probe is detected by a single-base extension reaction using the polymerase enzyme and dideoxynucleotides fluorescently labeled with 4 different dyes. (4) Reporter probes are separated from the template DNA

* Work supported in part by a "Large Grant" from the University of Connecticut's Research Foundation. A preliminary version of this manuscript appeared in [8].

and hybridized to the universal array. (5) Finally, fluorescence levels are used to determine which primers have been extended and learn the identity of the extending dideoxynucleotides.

In this paper we address two optimization problems arising in the design of genomic assays based on the universal tag arrays. First, we address the universal array *tag set design problem* (Section 2). To enable the economies of scale afforded by high-volume production of the arrays, tag sets must be designed to work well for a wide range of assay types and experimental conditions. Ben Dor et al. [2] have previously formalized the problem by imposing constraints on antitag-to-tag hybridization specificity under a hybridization model based on the classical 2-4 rule [10]. We extend the model in [2] to also prevent antitag-to-antitag hybridization and the formation of antitag secondary structures, which can significantly interfere with or disrupt correct assay functionality. Our results on this problem include a constructive upper bound on the maximum number of tags satisfying the extended constraints, as well as a simple alphabetic tree search tag selection algorithm.

Second, we study methods for improving the multiplexing rate (defined as the average number of reactions assayed per array) in large-scale genomic assays involving multiple universal arrays. In general, it is not possible to assign all tags to primers in an array experiment due to, e.g., unwanted primer-to-tag hybridizations. An assay specific optimization that determines the multiplexing rate (and hence the number of required arrays for a large assay) is the *tag assignment problem*, whereby individual (anti)tags are assigned to each primer. In Section 3 we observe that significant improvements in multiplexing rate can be achieved by combining primer selection with tag assignment. For most universal array applications there are multiple primers with the desired functionality; for example in the SNP genotyping assay described above one can choose the primer from either the forward or reverse strands. Since different primers hybridize to different sets of tags, a higher multiplexing rate is achieved by integrating primer selection with tag assignment. This integrated optimization is shown in Section 4 to lead to a reduction of up to 50% in the number of required arrays.

2 Universal Array Tag Set Design

The main objective of universal array tag set design is to maximize the number of tags, which directly determines the number of reactions that can be multiplexed using a single array. Tags are typically required to have a predetermined length [1, 7]. Furthermore, for correct assay functionality, tags and their antitags must satisfy the following hybridization constraints:

(H1) Every antitag hybridizes strongly to its tag;
(H2) No antitag hybridizes to a tag other than its complement; and
(H3) There is no antitag-to-antitag hybridization (including hybridization between two copies of the same tag and self-hybridization), since the formation of such duplexes and hair-pin structures prevents corresponding reporter probes from hybridizing to the template DNA and/or leads to undesired primer mis-extensions.

Hybridization affinity between two oligonucleotides is commonly characterized using the *melting temperature*, defined as the temperature at which 50% of the duplexes are in hybridized state. As in previous works [2, 3], we adopt a simple hybridization model to formalize constraints (H1)-(H3). This model is based on the observation that stable hybridization requires the formation of an initial *nucleation complex* between two perfectly complementary substrings of the two oligonucleotides. For such complexes, hybridization affinity is well approximated using the classical *2-4 rule* [10], which estimates the melting temperature of the duplex formed by an oligonucleotide with its complement as the sum between twice the number of A+T bases and four times the number of G+C bases.

The *complement* of a string $x = a_1 a_2 \dots a_k$ over the DNA alphabet $\{\mathtt{A}, \mathtt{C}, \mathtt{T}, \mathtt{G}\}$ is $\bar{x} = b_1 b_2 \dots b_k$, where b_i is the Watson-Crick complement of a_{k-i+1}. The *weight* $w(x)$ of x is defined as $w(x) = \sum_{i=1}^{k} w(a_i)$, where $w(\mathtt{A}) = w(\mathtt{T}) = 1$ and $w(\mathtt{C}) = w(\mathtt{G}) = 2$.

Definition 1. *For given constants l, h, and c with $l \leq h \leq 2l$, a set of tags $\mathcal{T} \subseteq \{\mathtt{A}, \mathtt{C}, \mathtt{T}, \mathtt{G}\}^l$ is called* feasible *if the following conditions are satisfied:*

- *(C1) Every tag in $\mathcal{T}$ has weight h or more.*
- *(C2) Every DNA string of weight c or more appears as substring at most once in the tags of $\mathcal{T}$.*
- *(C3) If a DNA string x of weight c or more appears as a substring of a tag, then $\bar{x}$ does not appear as a substring of a tag unless $x = \bar{x}$.*

The constants l, h, and c depend on factors such as array manufacturing technology and intended hybridization conditions. Property (H1) is implied by (C1) when h is large enough. Similarly, properties (H2) and (H3) are implied by (C1) and (C2) when c is small enough: constraint (C2) ensures that nucleation complexes do not form between antitags and non-complementary tags, while constraint (C3) ensures that nucleation complexes do not form between pairs of antitags.

Universal Array Tag Set Design Problem: *Given constants l, h, and c with $l \leq h \leq 2l$, find a feasible tag set of maximum cardinality.*

Ben-Dor et al. [2] have recently studied a simpler formulation of the problem in which tags of unequal length are allowed and only constraints (C1) and (C2) are enforced. For this simpler formulation, Ben-Dor et al. established a constructive upperbound on the optimal number of tags, and gave a nearly optimal tag selection algorithm based on De Bruijn sequences. Here, we refine the techniques in [2] to establish a constructive upperbound on the number of tags of a feasible set for the extended problem formulation, and propose a simple alphabetic tree search algorithm for constructing feasible tag sets.

The constructive upperbound is based on counting the minimal strings, called *c-tokens*, that can occur as substrings only once in the tags and antitags of a feasible set. Formally, a DNA string x is called c-token if the weight of x is c or more, and every proper suffix of x has weight strictly less than c. The *tail weight* of a c-token is defined as the weight of its last letter. Note that the weight of a c-token can be either c or $c+1$, the latter case being possible only if the c-token

starts with a G or a C. As in [2], we use G_n to denote the number of DNA strings of weight n. It is easy to see that $G_1 = 2$, $G_2 = 6$, and $G_n = 2G_{n-1} + 2G_{n-2}$; for convenience, we also define $G_0 = 1$. In Appendix A we prove the following:

Lemma 1. *Let $c \geq 4$. Then the total number of c-tokens that appear as substrings in a feasible tag set is at most $3G_{c-2} + 6G_{c-3} + G_{\frac{c-3}{2}}$ if c is odd, and at most $3G_{c-2} + 6G_{c-3} + \frac{1}{2}G_{\frac{c}{2}}$ if c is even. Furthermore, the total tail weight of c-tokens that appear as substrings in a feasible tag set is at most $2G_{c-1} + 4G_{c-3} + 2G_{\frac{c-3}{2}}$ if c is odd, and at most $2G_{c-1} + 4G_{c-3} + G_{\frac{c-2}{2}} + 2G_{\frac{c-4}{2}}$ if c is even.*

Theorem 1. *For every l, h, c with $l \leq h \leq 2l$ and $c \geq 4$, the number of tags in a feasible tag set is at most*

$$\min\left\{\frac{3G_{c-2} + 6G_{c-3} + G_{\frac{c-3}{2}}}{l-c+1}, \frac{2G_{c-1} + 4G_{c-3} + 2G_{\frac{c-3}{2}}}{h-c+1}\right\}$$

for c odd, and at most

$$\min\left\{\frac{3G_{c-2} + 6G_{c-3} + \frac{1}{2}G_{\frac{c}{2}}}{l-c+1}, \frac{2G_{c-1} + 4G_{c-3} + G_{\frac{c-2}{2}} + 2G_{\frac{c-4}{2}}}{h-c+1}\right\}$$

for c even.

Proof. The proof follows from Lemma 1 by observing that every tag contains at least $l - c + 1$ c-tokens, with a total tail weight of at least $h - c + 1$. □

To generate feasible sets of tags we employ a simple alphabetic tree search algorithm (see Figure 1). A similar algorithm is suggested in [7] for the problem of finding sets of tags that satisfy an unweighted version of constraint (C2). We start with an empty set of tags and an empty tag prefix. In every step we try to extend the current tag prefix t by an additional A. If the added letter completes a c-token or a complement of a c-token that has been used in already selected tags or in t itself, we try the next letter in the DNA alphabet, or backtrack to a previous position in the prefix when no more letter choices are left. Whenever we succeed generating a complete tag, we save it and backtrack to the last letter of its first c-token.

3 Improved Multiplexing by Integrated Primer Selection and Tag Assignment

Although constraints (H2)-(H3) in Section 2 prevent unintended antitag-to-tag and antitag-to-antitag hybridizations, the formation of nucleation complexes involving (portions of) the primers may still lead to undesired hybridization between reporter probes and tags on the array (Figure 2(a)), or between two reporter probes (Figure 2(b)-(d)). The formation of these duplexes must be avoided

```
Input: Positive integers c and l, c ≤ l
Output: Feasible MTSDP(l|C|1) solution 𝒯
---
Mark all c-tokens as available
For every i ∈ {1, 2, ..., l}, B_i ← A
𝒯 ← ∅; Finished ← 0; pos ← 1
While Finished = 0 do
    While the weight of B_1 B_2 ... B_pos < c do
        pos ← pos + 1
    EndWhile
    If the c-token ending B_1 B_2 ... B_pos is available then
        Mark the c-token ending at position pos as unavailable
        If pos = l then
            𝒯 ← 𝒯 ∪ {B_1 B_2 ... B_l}
            pos ← [the position where the first c-token of B_1 B_2 ... B_l ends]
            I ← {i | 1 ≤ i ≤ pos, B_i ≠ G}
            If I = ∅ then
                Finished ← 1
            Else
                pos ← max{I}
                B_i ← A for all i ∈ {pos + 1, ..., l}
                B_pos ← nextbase(B_pos)
            EndIf
        Else
            pos ← pos + 1
        EndIf
    Else
        I ← {i | 1 ≤ i ≤ pos, B_i ≠ G}
        If I = ∅ then
            Mark all the c-tokens in B_1 B_2 ... B_pos−1 as available
            Finished ← 1
        Else
            prevpos ← pos
            pos ← max{I}
            Mark all the c-tokens in B_pos ... B_prevpos−1 as available
            B_i ← A for all i ∈ {pos + 1, ..., l}
            B_pos ← nextbase(B_pos)
        EndIf
    EndIf
EndWhile
```

Fig. 1. The alphabetic tree search algorithm for MTSDP($l|C|1$). The *nextbase*$(\cdot)$ function is defined by *nextbase*(A) = T, *nextbase*(T) = C, and *nextbase*(C) = G.

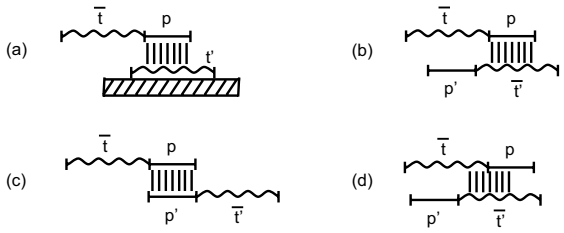

Fig. 2. Four types of undesired hybridizations, caused by the formation of nucleation complexes between (a) a primer and a tag other than the complement of the ligated antitag, (b) a primer and an antitag, (c) two primers, and (d) two reporter probe substrings, at least one of which straddles a ligation point.

as it leads to extension misreporting, false primer extensions, and/or reduced effective reporter probe concentration available for hybridization to the template DNA or to the tags on the array [3]. This can be done by leaving some of the tags unassigned. As in [3], we focus on preventing primer-to-tag hybridizations (Figure 2(a)). Our algorithms can be easily extended to prevent primer-to-antitag hybridizations (Figure 2(b)); a simple practical solution for preventing the other (less-frequent) unwanted hybridizations is to re-assign offending primers in a post-processing step.

Following [3], a set $\mathcal{P}$ of primers is called *assignable* to a set $\mathcal{T}$ of tags if there is a one-to-one mapping $a : \mathcal{P} \rightarrow \mathcal{T}$ such that, for every tag t hybridizing to a primer $p \in \mathcal{P}$, either $t \notin a(\mathcal{P})$ or $t = a(p)$.

Universal Array Multiplexing Problem: *Given primers $\mathcal{P} = \{p_1, \ldots, p_m\}$ and tag set $\mathcal{T} = \{t_1, \ldots, t_n\}$, find a partition of $\mathcal{P}$ into the minimum number of assignable sets.*

For most universal array applications there are multiple primers with the desired functionality, e.g., for the SNP genotyping assay described in Section 1, one can choose the primer from either the forward or reverse strands. Since different primers have different hybridization patterns, a higher multiplexing rate can in general be achieved by integrating primer selection with tag assignment. A similar integration has been recently proposed in [6] between probe selection and physical DNA array design, with the objective of minimizing unintended illumination in photo-lithographic manufacturing of DNA arrays. The idea in [6] is to modify probe selection tools to return *pools* containing all feasible candidates, and let subsequent optimization steps select the candidate to be used from each pool. In this paper we use a similar approach. We say that a set of primer pools is *assignable* if we can select a primer from each pool to form an assignable set of primers.

Pooled Universal Array Multiplexing Problem: *Given primer pools $\mathcal{P} = \{P_1, \ldots, P_m\}$ and tag set $\mathcal{T} = \{t_1, \ldots, t_n\}$, find a partition of $\mathcal{P}$ into the minimum number of assignable sets.*

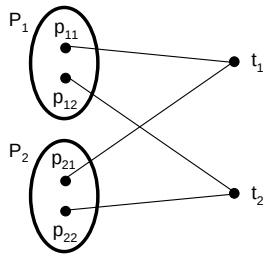

Fig. 3. Two assignable pools for which $|X(\mathcal{P})| + |Y(\mathcal{P})| = 0$.

Input: Primer pools $\mathcal{P} = \{P_1, \ldots, P_m\}$ and tag set $\mathcal{T}$
Output: Triples (p_i, t_i, k_i), $1 \leq i \leq m$, where $p_i \in P_i$ is the selected primer for pool i, t_i is the tag assigned to p_i, and k_i is the index of the array on which p_i is assayed

---

$k \leftarrow 0$
While $\mathcal{P} \neq \emptyset$ **do**
  $k \leftarrow k + 1$; $\mathcal{P}' \leftarrow \mathcal{P}$
  While $|X(\mathcal{P}')| + |Y(\mathcal{P}')| < |\mathcal{P}'|$ **do**
    Remove the primer p of maximum potential from the pools in $\mathcal{P}'$
    If p's pool becomes empty **then** remove it from $\mathcal{P}'$
  End While
  Assign pools in $\mathcal{P}'$ to tags on array k
  $\mathcal{P} \leftarrow \mathcal{P} \setminus \mathcal{P}'$
End While

Fig. 4. The iterative primer deletion algorithm.

Let $\mathcal{P}$ be a set of primer pools and $\mathcal{T}$ a tag set. For a primer p (tag t), $\mathcal{T}(p)$ (resp. $\mathcal{P}(t)$) denotes the set of tags (resp. primers of $\bigcup_{P \in \mathcal{P}} P$) hybridizing with p (resp. t). Let $X(\mathcal{P}) = \{P \in \mathcal{P} \ : \ \exists p \in P, \ t \in \mathcal{T} \text{ s.t. } t \in \mathcal{T}(p) \text{ and } \mathcal{P}(t) \subseteq P\}$ and $Y(\mathcal{P}) = \{t \in \mathcal{T} \ : \ \mathcal{P}(t) = \emptyset\}$. Clearly, in every pool of $X(\mathcal{P})$ we can find a primer p that hybridizes to a tag t which is not cross-hybridizing to primers in other pools, and therefore assigning t to p will not violate (A1). Furthermore, any primer can be assigned to a tag in $Y(\mathcal{P})$ without violating (A1). Thus, a set $\mathcal{P}$ with $|X(\mathcal{P})| + |Y(\mathcal{P})| \geq |\mathcal{P}|$ is always assignable. The converse is not necessarily true: Figure 3 shows two pools that are assignable although $|X(\mathcal{P})| + |Y(\mathcal{P})| = 0$.

Our primer pool assignment algorithm (see Figure 4) is a generalization to primer pools of Algorithm B in [3]. In each iteration, the algorithm checks whether $|X(\mathcal{P}')| + |Y(\mathcal{P}')| \geq |\mathcal{P}'|$ for the remaining set of pools $\mathcal{P}'$. If not, a primer of maximum *potential* is deleted from the pools. As in [3], the potential of a tag t with respect to $\mathcal{P}'$ is $2^{-|\mathcal{P}'(t)|}$, and the potential of a primer p is the sum of potentials for the tags in $\mathcal{T}(p)$. If the algorithm deletes the last primer in a pool P, then P is itself deleted from $\mathcal{P}'$; deleted pools are subsequently assigned to new arrays using the same algorithm.

Table 1. Tag sets selected by the alphabetic tree search algorithm.

l	$h_{min}/$ h_{max}	c	(C1)+(C2)				(C1)+(C2)+(C3)			
			tags	Bound	c-tokens	Bound	tags	Bound	c-tokens	Bound
20	–/–	8	213	275	2976	3584	107	132	1480	1726
		9	600	816	7931	9792	300	389	3939	4672
		10	1667	2432	20771	26752	844	1161	10411	12780
–	28/32	8	175	224	2918	3584	90	109	1489	1726
		9	531	644	8431	9792	263	312	4158	4672
		10	1428	1854	21707	26752	714	896	10837	12780
20	28/32	8	108	224	1548	3584	51	109	703	1726
		9	333	644	4566	9792	164	312	2185	4672
		10	851	1854	11141	26752	447	896	5698	12780

4 Experimental Results

Tag Set Selection. The alphabetic tree search algorithm described in Section 2 can be used to fully or selectively enforce the constraints in Definition 1. In order to assess the effect of various hybridization constraints on tag set size, we ran the algorithm both with constraints (C1)+(C2) and with constraints (C1)+(C2)+(C3). For each set of constraints, we ran the algorithm with c between 8 and 10 for typical practical requirements [1, 7] that all tags have length 20 and weight between 28 and 32 (corresponding to a GC-content between 40-60%). We also ran the algorithm with the tag length and weight requirements enforced individually.

Table 1 gives the size of the tag set found by the alphabetic tree search algorithm, as well as the number of c-tokens appearing in selected tags. We also include the theoretical upper-bounds on these two quantities; the upper-bounds for (C1)+(C2) follow from results of [2], while the upper-bounds for (C1)+(C2)+(C3) follow from Lemma 1 and Theorem 1. The results show that, for any combination of length and weight requirements, imposing the antitag-to-antitag hybridization constraints (C3) roughly halves the number of tags selected by the alphabetic tree search algorithm – as well as the theoretical upperbound – compared to only imposing antitag-to-tag hybridization constraints (C1)+(C2). For a fixed set of hybridization constraints, the largest tag sets are found by the alphabetic tree search algorithm when only the length requirement is imposed. The tag weight requirement, which guarantees similar melting temperatures for the tags, results in a 10-20% reduction in the number of tags. However, requiring that the tags have *both* equal length and similar weight results in close to halving the number of tags. This strongly suggests reassessing the need for the strict simultaneous enforcement of the two constraints in current industry designs [1]; our results indicate that allowing small variations in tag length and/or weight results in significant increases in the number of tags.

Integrated Primer Selection and Tag Assignment. We have implemented the iterative primer deletion algorithm in Figure 4 (Primer-Del), a variant of it

in which primers in pools of size 1 are omitted – unless all pools have size 1 – when selecting the primer with maximum potential for deletion (Primer-Del+), and two simple heuristics that first select from each pool the primer of minimum potential (Min-Pot), respectively minimum degree (Min-Deg), and then run the iterative primer deletion algorithm on the resulting pools of size 1. We ran all algorithms on data sets with between 1000 to 5000 pools of up to 5 randomly generated primers. As in [3], we varied the number of tags between 500 and 2000.

For instance size, we report the number of arrays and the average tag utilization (computed over all arrays except the last) obtained by (a) algorithm B in [3] run using a single primer per pool, (b) the four pool-aware assignment algorithms run with 1 additional candidate in each pool, and (c) the four pool-aware assignment algorithms run with 4 additional candidates in each pool. Scenario (b) models SNP genotyping applications in which the primer can be selected from both strands of the template DNA, while scenario (c) models applications such as gene transcription monitoring, where significantly more than 2 gene specific primers are typically available.

In a first set of experiments we extracted tag sequences from the tag set of the commercially available GenFlex Tag Arrays. All GenFlex tags have length 20; primers used in our experiments are 20 bases long as well. Primer-to-tag hybridizations were assumed to occur between primers and tags containing complementary c-tokens with $c = 7$ (Table 2), respectively $c = 8$ (Table 3). The results show that significant improvements in multiplexing rate – and a corresponding reduction in the number of arrays – are achieved by the pool-aware algorithms over the algorithm in [3]. For example, assaying 5000 reactions on a 2000-tag array requires 18 arrays using the method in [3] for $c = 7$, compared to only 13 (respectively 9) if 2 (respectively 5) primers per pool are available. In these experiments, the Primer-Del+ algorithm dominates in solution quality the Primer-Del, while Min-Deg dominates Min-Pot. Neither Primer-Del+ nor Min-Deg consistently outperforms the other over the whole range of parameters, which suggests that a good practical meta-heuristic is to run both of them and pick the best solution obtained.

In a second set of experiments we compared two sets of 213 tags of length 20, one constructed by running the alphabetic tree search algorithm in Section 2 with $c = 8$ and constraints (C1)+(C2), and the other extracted from the GenFlex Tag Array. The results in Table 4 show that the tags selected by the alphabetic tree search algorithm participate in fewer primer-to-tag hybridizations, which leads to an improved multiplexing rate.

References

1. Affymetrix, Inc. GeneFlex tag array technical note no. 1, available online at http://www.affymetrix.com/support/technical/technotes/genflex_technote.pdf.
2. A. Ben-Dor, R. Karp, B. Schwikowski, and Z. Yakhini. Universal DNA tag systems: a combinatorial design scheme. *Journal of Computational Biology*, 7(3-4):503–519, 2000.
3. A. BenDor, T. Hartman, B. Schwikowski, R. Sharan, and Z. Yakhini. Towards optimally multiplexed applications of universal DNA tag systems. In *Proc. 7th*

Annual International Conference on Research in Computational Molecular Biology, pages 48–56, 2003.
4. S. Brenner. Methods for sorting polynucleotides using oligonucleotide tags. *US Patent 5,604,097*, 1997.
5. J.N. Hirschhorn et al. SBE-TAGS: An array-based method for efficient single-nucleotide polymorphism genotyping. *PNAS*, 97(22):12164–12169, 2000.
6. A.B. Kahng, I.I. Măndoiu, S. Reda, X. Xu, and A. Zelikovsky. Design flow enhancements for DNA arrays. In *Proc. IEEE International Conference on Computer Design (ICCD)*, pages 116–123, 2003.
7. M.S. Morris, D.D. Shoemaker, R.W. Davis, and M.P. Mittmann. Selecting tag nucleic acids. *U.S. Patent 6,458,530 B1*, 2002.
8. I.I. Măndoiu, C. Prăjescu, and D. Trincă. Improved tag set design and multiplexing algorithms for universal arrays. In V.S. Sunderam et al., editor, *Proc. IWBRA 2005/ICCS 2005*, volume 3515 of *Lecture Notes in Computer Science*, pages 994–1002, Berlin, 2005. Springer-Verlag.
9. N.P. Gerry et al. Universal DNA microarray method for multiplex detection of low abundance point mutations. *J. Mol. Biol.*, 292(2):251–262, 1999.
10. R.B. Wallace, J. Shaffer, R.F. Murphy, J. Bonner, T. Hirose, and K. Itakura. Hybridization of synthetic oligodeoxyribonucleotides to phi chi 174 DNA: the effect of single base pair mismatch. *Nucleic Acids Res.*, 6(11):6353–6357, 1979.

A Proof of Lemma 1

We first establish two lemmas on self-complementary DNA strings, i.e., strings $x \in \{\mathtt{A}, \mathtt{C}, \mathtt{T}, \mathtt{G}\}^+$ with $x = \overline{x}$.

Lemma 2. *If x is self-complementary then $|x|$ and $w(x)$ are both even.*

Proof. Let $x = x_1 x_2 \ldots x_p$ be a self-complementary DNA string. If $p = 2q + 1$, by the definition of the complement we should have $x_{q+1} = \overline{x}_{q+1}$, which is impossible. Thus, $p = 2q$. Since $x_1 = \overline{x}_{2q}$,$x_2 = \overline{x}_{2q-1}$,..., $x_q = \overline{x}_{q+1}$, and the weight of complementary bases is the same, it follows that $w(x) = 2\sum_{i=1}^{q} w(x_i)$. □

Lemma 3. *Let H_n be the number of self-complementary DNA strings of weight n. $H_n = 0$ if n is odd, and $H_n = G_{n/2}$ if n is even.*

Proof. By Lemma 2, self-complementary strings must have even length and weight. For even n, the mapping $x_1 \ldots x_q x_{q+1} \ldots x_{2q} \mapsto x_1 \ldots x_q$ gives a one-to-one correspondence between self-complementary strings of weight n and strings of weight $n/2$. □

Proof of Lemma 1. Let `W` and `S` denote weak and strong DNA bases (`A` or `T`, respectively `G` or `C`), and let $<w>$ denote the set of DNA strings with weight w. The c-tokens can be partitioned into the seven classes given in Table 5, depending on total token weight (c or $c+1$) and the type of starting and ending bases. This partitioning is defined so that, for every c-token x, the class of the unique c-token suffix of $\overline{x}$ can be determined from the class of x. Note that $\bar{x}$ is itself a c-token, except when $x \in \mathtt{S}<c-3>\mathtt{WW} \cup \mathtt{S}<c-4>\mathtt{SW}$.
Let N_{cls} denote the number of c-tokens of class *cls* occurring in a feasible tag set.

Table 2. Multiplexing results for $c = 7$ (averages over 10 test cases).

# pools	Pool size	Algorithm	500 tags		1000 tags		2000 tags	
			#arrays	% Util.	#arrays	% Util.	#arrays	% Util.
1000	1	[3]	7.5	30.1	6.0	19.3	5.0	12.1
	2	Primer-Del	6.0	38.7	5.0	24.3	4.1	15.5
	2	Primer-Del+	6.0	39.6	4.5	27.3	4.0	16.5
	2	Min-Pot	6.0	38.4	5.0	24.2	4.0	15.9
	2	Min-Deg	5.8	40.9	4.6	27.0	4.0	16.4
	5	Primer-Del	5.0	49.6	4.0	32.5	3.3	21.0
	5	Primer-Del+	4.0	60.4	3.0	43.6	3.0	24.7
	5	Min-Pot	4.9	50.6	4.0	33.0	3.0	23.5
	5	Min-Deg	4.0	62.0	3.0	44.9	2.7	28.1
2000	1	[3]	13.4	31.8	11.0	19.9	8.7	12.9
	2	Primer-Del	10.7	41.0	8.5	26.4	7.0	16.6
	2	Primer-Del+	10.0	43.3	8.0	28.1	6.0	19.1
	2	Min-Pot	11.0	39.4	9.0	24.8	7.0	16.3
	2	Min-Deg	10.0	43.5	8.0	28.2	6.0	19.2
	5	Primer-Del	8.0	56.8	6.1	38.4	5.0	24.5
	5	Primer-Del+	7.1	62.4	6.0	39.7	4.0	30.1
	5	Min-Pot	9.2	47.5	7.0	32.9	5.0	24.0
	5	Min-Deg	7.0	63.1	5.3	44.2	4.0	30.7
5000	1	[3]	29.5	35.0	23.0	22.6	18.0	14.6
	2	Primer-Del	22.2	47.0	17.1	30.9	13.7	19.6
	2	Primer-Del+	22.2	46.8	17.0	30.9	13.1	20.4
	2	Min-Pot	25.0	41.5	19.2	27.3	15.0	17.7
	2	Min-Deg	22.0	47.3	17.0	31.0	13.0	20.6
	5	Primer-Del	16.6	63.8	12.3	43.9	10.0	27.8
	5	Primer-Del+	16.0	65.6	12.0	44.9	9.0	30.6
	5	Min-Pot	29.5	35.0	23.0	22.6	18.0	14.6
	5	Min-Deg	16.0	65.8	12.0	45.2	9.0	30.8

Table 3. Multiplexing results for $c = 8$ (averages over 10 test cases).

# pools	Pool size	Algorithm	500 tags #arrays	500 tags % Util.	1000 tags #arrays	1000 tags % Util.	2000 tags #arrays	2000 tags % Util.
1000	1	[3]	3.0	86.0	2.0	77.1	2.0	46.3
	2	Primer-Del	3.0	90.1	2.0	81.6	2.0	47.8
	2	Primer-Del+	3.0	94.5	2.0	88.5	1.0	50.0
	2	Min-Pot	3.0	94.4	2.0	87.9	1.0	50.0
	2	Min-Deg	3.0	92.6	2.0	88.8	1.0	50.0
	5	Primer-Del	3.0	98.0	2.0	92.6	2.0	49.2
	5	Primer-Del+	3.0	99.5	2.0	97.4	1.0	50.0
	5	Min-Pot	3.0	99.4	2.0	97.1	1.0	50.0
	5	Min-Deg	3.0	93.4	2.0	93.4	1.0	50.0
2000	1	[3]	6.0	78.2	4.0	64.4	3.0	48.3
	2	Primer-Del	5.0	92.3	4.0	66.6	3.0	49.8
	2	Primer-Del+	5.0	93.5	3.0	87.9	2.0	78.7
	2	Min-Pot	5.0	93.6	3.0	87.7	2.0	78.1
	2	Min-Deg	5.0	90.8	3.0	87.5	2.0	79.6
	5	Primer-Del	5.0	98.4	3.0	94.1	2.0	84.8
	5	Primer-Del+	5.0	99.5	3.0	97.1	2.0	91.2
	5	Min-Pot	5.0	99.5	3.0	97.0	2.0	90.8
	5	Min-Deg	5.0	91.8	3.0	90.6	2.0	91.7
5000	1	[3]	13.0	81.3	8.6	64.7	6.0	49.3
	2	Primer-Del	12.0	90.5	7.0	81.1	5.0	61.7
	2	Primer-Del+	11.2	93.8	7.0	81.9	4.0	73.8
	2	Min-Pot	12.0	90.4	7.0	81.2	5.0	62.2
	2	Min-Deg	12.0	90.1	7.0	81.5	4.0	73.9
	5	Primer-Del	11.0	98.9	6.0	96.1	4.0	81.7
	5	Primer-Del+	11.0	99.4	6.0	96.8	3.0	97.1
	5	Min-Pot	11.0	99.4	6.0	96.9	4.0	83.1
	5	Min-Deg	11.0	94.6	6.0	91.0	3.4	88.0

Table 4. Multiplexing results (averages over 10 test cases) for two sets of 213 tags of length 20, one constructed by running the alphabetic tree search algorithm in Section 2 with $c = 8$ and constraints (C1)+(C2), and the other extracted from the GenFlex Tag Array.

# pools	Pool size	Algorithm	GenFlex tags #arrays	GenFlex tags % Util.	Tree search tags #arrays	Tree search tags % Util.
1000	1	[3]	6.0	90.0	5.0	100.0
	2	Primer-Del+	5.0	100.0	5.0	100.0
	2	Min-Deg	5.9	94.0	5.0	100.0
	5	Primer-Del+	5.0	100.0	5.0	100.0
	5	Min-Deg	5.2	97.3	5.0	100.0
2000	1	[3]	11.0	90.6	10.0	99.2
	2	Primer-Del+	10.0	98.7	10.0	100.0
	2	Min-Deg	10.8	94.2	10.0	99.3
	5	Primer-Del+	10.0	100.0	10.0	100.0
	5	Min-Deg	10.1	96.0	10.0	99.3
5000	1	[3]	26.5	91.3	24.0	99.2
	2	Primer-Del+	25.0	97.6	24.0	100.0
	2	Min-Deg	25.0	96.3	24.0	99.3
	5	Primer-Del+	24.0	100.0	24.0	100.0
	5	Min-Deg	25.0	96.6	24.0	99.3

c odd

Since $\mathtt{W}\!<\!c-3\!>\!\mathtt{S} \cup \mathtt{S}\!<\!c-3\!>\!\mathtt{W}$ can be partitioned into $4G_{c-3}$ pairs $\{x, \bar{x}\}$ of complementary c-tokens, and at most one token from each pair can appear in a feasible tag set,

$$N_{\mathtt{W}<c-3>\mathtt{S}} + N_{\mathtt{S}<c-3>\mathtt{W}} \leq 4G_{c-3} \tag{1}$$

Similarly, class $\mathtt{W}\!<\!c-2\!>\!\mathtt{W}$ can be partitioned into $2G_{c-2}$ pairs $\{x, \bar{x}\}$ of complementary c-tokens, $\mathtt{W}\!<\!c-3\!>\!\mathtt{S} \cup \mathtt{S}\!<\!c-3\!>\!\mathtt{WW}$ can be partitioned into $4G_{c-3}$ triples $\{x, \bar{x}A, \bar{x}T\}$ with $x \in \mathtt{W}\!<\!c-3\!>\!\mathtt{S}$, $\mathtt{S}\!<\!c-3\!>\!\mathtt{W} \cup \mathtt{S}\!<\!c-3\!>\!\mathtt{WW}$ can be partitioned into $4G_{c-3}$ triples $\{x, xA, xT\}$ with $x \in \mathtt{S}\!<\!c-3\!>\!\mathtt{W}$, and $\mathtt{S}\!<\!c-4\!>\!\mathtt{S} \cup \mathtt{S}\!<\!c-4\!>\!\mathtt{SW}$ can be partitioned into $2G_{c-4}$ 6-tuples $\{x, \bar{x}, xA, xT, \bar{x}A, \bar{x}T\}$ with $x \in \mathtt{S}\!<\!c-4\!>\!\mathtt{S}$. Since at most one c-token can appear in a feasible tag set from each such pair,

Table 5. Classes of c-tokens.

Class of x	c-token suffix of $\overline{x}$
$\mathtt{W}\!<\!c-3\!>\!\mathtt{S}$	$\mathtt{S}\!<\!c-3\!>\!\mathtt{W}$
$\mathtt{S}\!<\!c-4\!>\!\mathtt{S}$	$\mathtt{S}\!<\!c-4\!>\!\mathtt{S}$
$\mathtt{S}\!<\!c-3\!>\!\mathtt{S}$	$\mathtt{S}\!<\!c-3\!>\!\mathtt{S}$
$\mathtt{W}\!<\!c-2\!>\!\mathtt{W}$	$\mathtt{W}\!<\!c-2\!>\!\mathtt{W}$
$\mathtt{S}\!<\!c-3\!>\!\mathtt{W}$	$\mathtt{W}\!<\!c-3\!>\!\mathtt{S}$
$\mathtt{S}\!<\!c-3\!>\!\mathtt{WW}$	$\mathtt{W}\!<\!c-3\!>\!\mathtt{S}$
$\mathtt{S}\!<\!c-4\!>\!\mathtt{SW}$	$\mathtt{S}\!<\!c-4\!>\!\mathtt{S}$

triple, respectively 6-tuple,

$$N_{\mathtt{W}<c-2>\mathtt{W}} \leq 2G_{c-2} \tag{2}$$

$$N_{\mathtt{W}<c-3>\mathtt{S}} + N_{\mathtt{S}<c-3>\mathtt{WW}} \leq 4G_{c-3} \tag{3}$$

$$N_{\mathtt{S}<c-3>\mathtt{W}} + N_{\mathtt{S}<c-3>\mathtt{WW}} \leq 4G_{c-3} \tag{4}$$

$$N_{\mathtt{S}<c-4>\mathtt{S}} + N_{\mathtt{S}<c-4>\mathtt{SW}} \leq 2G_{c-4} \tag{5}$$

Using Lemma 3, it follows that $\mathtt{S}<c-3>\mathtt{S}$ contains $2G_{\frac{c-3}{2}}$ self-complementary c-tokens. Since the remaining $4G_{c-3} - 2G_{\frac{c-3}{2}}$ c-tokens can be partitioned into complementary pairs each contributing at most one c-token to a feasible tag set,

$$N_{\mathtt{S}<c-3>\mathtt{S}} \leq \frac{1}{2}\left(4G_{c-3} - 2G_{\frac{c-3}{2}}\right) + 2G_{\frac{c-3}{2}} = 2G_{c-3} + G_{\frac{c-3}{2}} \tag{6}$$

Adding inequalities (1), (3), and (4) multiplied by 1/2 with (2), (5), and (6) implies that the total number of c-tokens in a feasible tag set is at most

$$2G_{c-2} + 8G_{c-3} + 2G_{c-4} + G_{\frac{c-3}{2}} = 3G_{c-2} + 6G_{c-3} + G_{\frac{c-3}{2}}$$

Furthermore, adding (1), (2), and (3) with inequalities (5) and (6) multiplied by 2 implies that the total tail weight of the c-tokens in a feasible tag set is at most

$$2G_{c-2} + 12G_{c-3} + 4G_{c-4} + 2G_{\frac{c-3}{2}} = 2G_{c-1} + 4G_{c-3} + 2G_{\frac{c-3}{2}}$$

***c* even**

Inequalities (1), (3), and (4) continue to hold for even values of c. Since $c-3$ is odd, $\mathtt{S}<c-3>\mathtt{S}$ contains no self-complementary tokens and can be partitioned into $2G_{c-3}$ pairs $\{x, \bar{x}\}$,

$$N_{\mathtt{S}<c-3>\mathtt{S}} \leq 2G_{c-3} \tag{7}$$

By Lemma 3, there are $2G_{\frac{c-4}{2}}$ self-complementary tokens in $\mathtt{S}<c-4>\mathtt{S}$. Therefore $\mathtt{S}<c-4>\mathtt{S} \cup \mathtt{S}<c-4>\mathtt{SW}$ can be partitioned into $2G_{\frac{c-4}{2}}$ triples $\{x, xA, xT\}$ with $x \in \mathtt{S}<c-4>\mathtt{S}$, $x = \bar{x}$ and $2G_{c-4} - G_{\frac{c-4}{2}}$ 6-tuples $\{x, \bar{x}, xA, xT, \bar{x}A, \bar{x}T\}$ with $x \in \mathtt{S}<c-4>\mathtt{S}$, $x \neq \bar{x}$. Since a feasible tag set can use at most one c-token from each triple and 6-tuple,

$$N_{\mathtt{S}<c-4>\mathtt{S}} + N_{\mathtt{S}<c-4>\mathtt{SW}} \leq 2G_{c-4} + G_{\frac{c-4}{2}} \tag{8}$$

Using again Lemma 3, we get

$$N_{\mathtt{W}<c-2>\mathtt{W}} \leq 2G_{c-2} + G_{\frac{c-2}{2}} \tag{9}$$

Adding inequalities (1), (3), and (4) multiplied by 1/2 with (7), (8), and (9) implies that the total number of c-tokens in a feasible tag set is at most

$$2G_{c-2} + 8G_{c-3} + 2G_{c-4} + G_{\frac{c-2}{2}} + G_{\frac{c-4}{2}} = 3G_{c-2} + 6G_{c-3} + \frac{1}{2}G_{\frac{c}{2}}$$

Finally, adding (1), (3), and (9) with inequalities (7) and (8) multiplied by 2 implies that the total tail weight of the c-tokens in a feasible tag set is at most

$$2G_{c-2} + 12G_{c-3} + 4G_{c-4} + G_{\frac{c-2}{2}} + 2G_{\frac{c-4}{2}} = 2G_{c-1} + 4G_{c-3} + G_{\frac{c-2}{2}} + 2G_{\frac{c-4}{2}}$$

□