Exact and Approximation Algorithms for DNA Tag Set Design*

Ion I. Măndoiu and Dragoş Trincă

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

Abstract. In this paper we propose new solution methods for designing tag sets for use in universal DNA arrays. First, we give integer linear programming formulations for two previous formalizations of the tag set design problem, and show that these formulations can be solved to optimality for instance sizes of practical interest by using general purpose optimization packages. Second, we note the benefits of periodic tags, and establish an interesting connection between the tag design problem and the problem of packing the maximum number of vertex-disjoint directed cycles in a given graph. We show that combining a simple greedy cycle packing algorithm with a previously proposed alphabetic tree search strategy yields an increase of over 40% in the number of tags compared to previous methods.

1 Introduction

Recently developed *universal DNA tag arrays* [5, 11, 14] offer a flexible and cost-effective alternative to custom-designed DNA arrays for performing a wide range of genomic analyses. A universal tag array consists of a set of DNA strings called *tags*, designed such that each tag hybridizes strongly to its own *antitag* (Watson-Crick complement), but to no other antitag. A typical assay based on universal tag arrays performs Single Nucleotide Polymorphism (SNP) genotyping using the following steps [3, 7]: (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. (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 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.

* Work supported in part by a Large Grant from the University of Connecticut's Research Foundation.

Tag set design involves balancing two conflicting requirements: on one hand we would like a large number of tags to allow assaying a large number of biochemical reactions, on the other hand we would like the tags 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, and have proposed near-optimal heuristics. In Section 3 we give an integer linear programming (ILP) formulation for this problem and its variant in which tags are required to have equal length [12]. Empirical results in Section 5 show that these ILP formulations have extremely small integrality gap, and can be solved to optimality for instance sizes of practical interest by using general purpose optimization packages.

Previous works on tag set design [2, 12] require for substrings that may form a nucleation complex and initiate cross hybridization not to be repeated within any selected tag. This constraint simplifies analysis, but is *not* required for ensuring correct tag functionality – what is required is for such substrings not to appear simultaneously in two different tags. To our knowledge, no previous work has assessed the impact that adding this constraint has on tag set size. In this paper we propose two algorithms for designing tag sets while relaxing this constraint. The first one is a modification of the alphabetic tree search strategy in [11, 12], The second algorithm stems from the observation that periodic tags, particularly those with a short period, use the least amount of "resources" and lead to larger tag sets, where the limited resources are in this case minimal substrings that can form nucleation complexes (for formal models see Section 2). In Section 4 we establish an interesting connection between the tag design problem and the problem of packing the maximum number of vertex-disjoint directed cycles in a given graph, and propose a simple greedy algorithm for the latter one. Results in Section 5 show that combining the greedy cycle packing algorithm with alphabetic tree search strategy yields an increase of over 40% in the number of tags compared to previous methods.

2 Problem Formulations and Previous Work

A main objective of universal array designers is to maximize the number of tags, which directly determines the number of reactions that can multiplexed using a single array. At the same time, tag sets must satisfy a number of *stability* and *non-interaction* constraints [4]. The set of constraints depends on factors such as the array manufacturing technology and the intended application. In this section we formalize the most im-

portant stability and non-interaction constraints using the hybridization model in [2].

Hybridization model. Hybridization affinity between two oligonucleotides is commonly characterized using the *melting temperature*, defined as the temperature at which half of the duplexes are in hybridized state and the other half are in melted state. However, accurate melting temperature estimation is computationally expensive, e.g., estimating the melting temperature between two non-complementary oligonucleotides using the near-neighbor model of SantaLucia [16] is an NP-hard problem [8]. Ben-Dor et al. [2, 3] formalized a conservative hybridization model 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 nucleation complexes, hybridization affinity is modeled using the classical *2-4 rule* [17], which estimates the melting temperature of the duplex formed by an oligonucleotide with its complement as the sum between the number of *weak* bases (i.e., A and T) and twice the number of *strong* bases (i.e., G and C).

The *weight* $w(x)$ of a DNA string $x = a_1 a_2 \dots a_k$ is defined as $w(x) = \sum_{i=1}^{k} w(a_i)$, where $w(\mathtt{A}) = w(\mathtt{T}) = 1$ and $w(\mathtt{C}) = w(\mathtt{G}) = 2$. Throughout this paper we assume the following *c-token hybridization model* [2]: hybridization between two oligonucleotides takes place only if one oligo contains as substring the complement of a substring of weight c or more of the other, where c is a given constant. 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 defined as $\bar{x} = b_1 b_2 \dots b_k$, where b_i is the Watson-Crick complement of a_{k-i+1}.

Hybridization stability. Current industry designs require a predetermined tag length l, e.g., GenFlex tag arrays manufactured by Affymetrix use $l = 20$ [1]. The model proposed in [2] allows tags of unequal length and instead require a minimum tag weight of h, for a given constant h. In this paper we consider both types of stability constraints, and use the parameter $\alpha \in \{l, h\}$ to denote the specific model used for hybridization stability.

Pairwise non-interaction constraints. A basic constraint in this category is for every antitag not to hybridize to non-complementary tags [2]. For a DNA string x and a set of tags $\mathcal{T}$, let $N_{\mathcal{T}}(x)$ denote the number of tags in $\mathcal{T}$ that contain x as a substring. Using the c-token hybridization model, the antitag-to-tag hybridization constraint is formalized as follows:

(C) For every feasible tag set $\mathcal{T}$, $N_{\mathcal{T}}(x) \leq 1$ for every DNA string x of weight c or more.

In many assays based on universal tag arrays it is also required to prevent antitag-to-antitag hybridization, since the formation of such antitag-to-antitag duplexes or antitag hair-pin structures prevents reporter probes from performing their function in the solution-based hybridization steps [4, 12]. The combined constraints on antitag hybridization are formalized as follows

($\bar{C}$) For every feasible tag set $\mathcal{T}$, $N_{\mathcal{T}}(x) + N_{\mathcal{T}}(\bar{x}) \leq 1$ for every DNA string x of weight c or more.

In the following we use the parameter $\beta \in \{C, \bar{C}\}$ to specify the type of pairwise hybridization constraints.

Substring occurrences within a tag. Previous works on DNA tag set design [2, 12] have imposed the following *c-token uniqueness constraint* in addition to constraints (C) and ($\bar{C}$): a DNA string of weight c or more can appear as a substring of a feasible tag at most once. This uniqueness constraint has been added purely for ease of analysis (e.g., it is the key property enabling the DeBruijn sequence based heuristics in [2]), and is *not* required for ensuring correct assay functionality. To our knowledge, no previous work has assessed the impact that adding this constraint has on tag set size. In the following we will use the parameter $\gamma \in \{1, multiple\}$ to specify whether or not the c-token uniqueness constraint is enforced.

For every $\alpha \in \{l, h\}$, $\beta \in \{C, \bar{C}\}$, and $\gamma \in \{1, multiple\}$, the maximum tag set design problem with constraints α, β, γ, denoted MTSDP($\alpha|\beta|\gamma$), is the following: given constants c and l/h, find a tag set of maximum cardinality satisfying the constraints.

Previous work on tag set design. Ben-Dor et al. [2] formalized the c-token model for oligonucleotide hybridization and studied the MTSDP($h|C|1$) problem. They established a constructive upperbound on the optimal number of tags for this formulation, and gave a nearly optimal tag selection algorithm based on DeBruijn sequences. Similar upper bounds are established for the MTSDP($l|C|1$) and MTSDP($*|\bar{C}|1$) problems in [12], which also extends a simple alphabetic tree search strategy originally proposed in [11] to handle all considered problem variants.

For a comprehensive survey of hybridization models, results on the associated formulations for the tag set design problem, and further motivating applications in the area of DNA computing, we direct the reader to [4].

3 Integer Linear Programming Formulations for MTSDP($*|C|1$)

Before stating our integer linear program formulation, we introduce some additional notations.

Following [2], a DNA string x of weight c or more is called a *c-token* if all its proper suffixes have weight strictly less than c. Clearly, it suffices to enforce constraint (C) for all c-tokens x. Let N denote the number of c-tokens, and $\mathcal{C} = \{c_1, \ldots, c_N\}$ denote the set of all c-tokens. The results in [2] imply that $N = \Theta((1+\sqrt{3})^c)$. 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 strong base (`G` or `C`). We let $\mathcal{C}_0 \subseteq \mathcal{C}$ denote the set of c-tokens of weight $c+1$ that end with a weak base, i.e., c-tokens of the form `S`$<c-2>$`W`, where `W` (`S`) denote a weak (strong) base, and $<c-2>$ denotes an arbitrary string of weight $c-2$. We also let $\mathcal{C}_2 \subseteq \mathcal{C}$ denote the set of c-tokens of weight c that end with a strong base, i.e., c-tokens of the form $<c-2>$`S`.

Clearly, there is at most one c-token ending at every letter of a tag. It is easy to see that each c-token $x \in \mathcal{C}_0$ contains a proper prefix which is itself a c-token, and therefore x cannot be the first c-token of a tag, i.e., cannot be the c-token with the leftmost ending. All other c-tokens can appear as first c-tokens. When a c-token in $C \setminus (\mathcal{C}_0 \cup \mathcal{C}_2)$ is the first in a tag, then it must be a prefix of the tag. On the other hand, tokens in $\mathcal{C}_2$ can be the first both in tags that they prefix and in tags in which they are preceded by a weak base not covered by any c-token.

The ILP formulation for MTSDP$(l|C|1)$ uses an auxiliary directed graph $G = (V, E)$ with $V = \{s, t\} \cup \bigcup_{1 \le i \le N} V_i$, where $V_i = \{v_i^k \mid |c_i| \le k \le l\}$. G has a directed arc from v_i^k to v_j^{k+1} for every triple i, j, k such that $|c_i| \le k \le l-1$ and c_j is obtained from c_i by appending a single nucleotide and removing the maximal prefix that still leaves a valid c-token. Finally, G has an arc from s to every $v \in V_{first}$, where $V_{first} = \{v_i^{|c_i|} \mid c_i \in \mathcal{C} \setminus \mathcal{C}_0\} \cup \{v_i^{|c_i|+1} \mid c_i \in \mathcal{C}_2\}$, and an arc from v_i^l to t for every $1 \le i \le N$.

We claim that, for $c \le l$, MTSDP$(l|C|1)$ can be reformulated as the problem of finding the maximum number of s-t paths in G that collectively visit at most one vertex v_i^k for every i. Indeed, let P be an s-t path and v_i^k be the vertex following s in P. If $k = |c_i|$, we associate to P the tag obtained by concatenating c_i with the last letters of the c-tokens corresponding to the subsequently visited vertices, until reaching t. Otherwise, if $k = |c_i| + 1$ (which implies that $c_i \in \mathcal{C}_2$) we associate to P the two tags obtained by concatenating either `A` or `T` with c_i and the last letters of subsequently visited c-tokens. The claim follows by observing that at most one of the tags associated with each path can be used in a feasible solution.

Our ILP formulation can be viewed as a generalized version of the integer maximum flow problem in which unit capacity constraints are imposed on *sets of vertices* of G instead of individual vertices. The for-

mulation uses 0/1 variables x_v and y_e for every every vertex $v \in V \setminus \{s,t\}$, respectively arc $e \in E$. These variables are set to 1 if the corresponding vertex or arc is visited by an s-t path corresponding to a selected tag. Let $in(v)$ and $out(v)$ denote the set of arcs entering, respectively leaving vertex v. The integer program can then be written as follows:

$$\text{maximize} \quad \sum_{v \in V_{first}} x_v \tag{1}$$

subject to

$$x_v = \sum_{e \in in(v)} y_e = \sum_{e \in out(v)} y_e, \quad v \in V \setminus \{s,t\} \tag{2}$$

$$\sum_{v \in V_i} x_v \leq 1, \qquad 1 \leq i \leq N \tag{3}$$

$$x_v, y_e \in \{0,1\}, \qquad v \in V \setminus \{s,t\}, e \in E \tag{4}$$

Constraints (2) ensure that variables y_e set to 1 correspond to a set of s-t paths, and that a variable x_v is set to 1 if and only if one of these paths passes through v.[1] Antitag-to-tag hybridization constraints (C) and c-token uniqueness are enforced by (3). Finally, the objective (1) corresponds to maximizing the number of selected tags, since the shortest prefix of a tag that is a c-token must belong to $\mathcal{C} \setminus \mathcal{C}_0$.

For a token $c_i = c_j X \in \mathcal{C}_0$, where $X \in \{\mathtt{A}, \mathtt{T}\}$, let $\widehat{c_i} = c_j \bar{a}$. Since both c_i and $\widehat{c_i}$ contain c_j as a prefix, and c_j can appear at most once in a feasible tag set $\mathcal{T}$, it follows that at most one of them can appear in $\mathcal{T}$. Therefore, the following valid inequality can be added to the the ILP formulation (1)–(4) to improve its integrality gap (i.e., the gap between the value of the optimum integer solution and that of the optimal fractional relaxation):

$$\sum_{v \in V_i \cup V_j} x_v \leq 1, \qquad c_i \in \mathcal{C}_0,\ c_j = \widehat{c_i},\ i < j \tag{5}$$

The formulation of MTSDP($l|C|1$) has exactly the same objective and constraints for a slightly different graph G. Let us define the *tail weight* of a c-token C, denoted $tail(C)$, as the weight of C's last letter. Also, let $h_i = h$ if c_i has a tail weight of 1 and $h_i = h + 1$ if c_i has a tail weight of 2. We will require that every tag ending with token c_i has total weight of at most h_i; it is easy to see that this constraint is not affecting the size of the optimum tag set. We now define the graph $G = (V, E)$ with $V = \{s,t\} \cup \bigcup_{1 \leq i \leq N} V_i$, where $V_i = \{v_i^k \mid w(c_i) \leq k \leq h_i\}$. G

[1] Variables x_v can be eliminated by replacing them with the corresponding sums of x_e's; we use them here merely for improving readability. ILP sizes reported in Section 5 refer to the equivalent reduced formulations obtained by eliminating these variables.

contains a directed arc from v_i^k to $v_j^{k+tail(i)}$ for every triple i, j, k such that $|c_i| \leq k \leq h_i - tail(c_i)$ and c_j is obtained from c_i by appending a single nucleotide and removing the maximal prefix that still leaves a valid c-token. Finally, G contains arcs from s to every $v \in V_{first}$, where V_{first} is now equal to $\{v_i^{w(c_i)} \mid c_i \in \mathcal{C} \setminus \mathcal{C}_0\} \cup \{v_i^{w(c_i)+1} \mid c_i \in \mathcal{C}_2\}$, plus arcs from every v_i^k to t for every $1 \leq i \leq N$ and $h_i - tail(c_i) < k \leq h_i$.

4 Algorithms for MTSDP($*|*|multiple$)

In the following we describe two algorithms for MTSDP($l|C|multiple$); both algorithms can be easily adjusted to handle the other MTSDP($*|*|multiple$) variants. The first algorithm (see [13] for a detailed pseudocode) is similar to the alphabetic tree search algorithms proposed for MTSDP($l|C|1$) in [12]. The algorithm performs an alphabetical traversal of a 4-ary tree representing all 4^l possible tags, skipping over subtrees rooted at internal vertices that correspond to tag prefixes including unavailable c-tokens. The difference from the MTSDP($l|C|1$) algorithm in [12] lies in the strategy used to mark c-tokens as unavailable. While the algorithm in [12] marks a c-token C as unavailable as soon as it incorporates it in the current tag prefix (changing C's status back to "available" when forced to backtrack past C's tail), our algorithm marks a c-token as unavailable only when a complete tag is found.

Note that the alphabetic tree search algorithm produces a *maximal* feasible set of tags $\mathcal{T}$, i.e., there is no tag t such that $\mathcal{T} \cup \{t\}$ remains feasible for MTSDP($l|C|multiple$). Hence, every tag of an optimal solution must share at least one c-token with tags in $\mathcal{T}$. Since every tag of $\mathcal{T}$ has at most $l - c/2 + 1$ c-tokens, it follows that the alphabetic tree algorithm (and indeed, every algorithm that produces a maximal feasible set of tags) has an approximation factor of $l - c/2 + 1$.

We call a tag t *periodic* if t is the length l prefix of an infinite string x^∞, where x is a DNA string with $|x| < |t|$. (Note that a periodic tag t is not necessarily the concatenation of an integer number of copies of its period x as in the standard definition of string periodicity [10].)

The following lemma shows that tag set design algorithms can restrict the search to two simple classes of tags.

Lemma 1. *For every c and l, there exists an optimal tag set $\mathcal{T}$ in which every tag has the uniqueness property or is periodic.*[2]

Proof. Let $\mathcal{T}$ be an optimal tag set. Assume that $\mathcal{T}$ contains a tag t that does not have the uniqueness property, and let $c_{i_1}, \ldots, c_{i_k}$ be the

[2] Note that the two classes of tags are not disjoint, since there exist periodic tags that have the uniqueness property.

sequence of c-tokens occurring in t, in left to right order. Since t does not have the uniqueness property, there exist indices $1 \leq j < j' \leq i_k$ such that $c_{i_j} = c_{i_{j'}}$. Let t' be the tag formed by taking the first l letters of the infinite string with c-token sequence $(c_{i_j}, \ldots, c_{i_{j'-1}})^\infty$; note that t' is a periodic tag. Since c-tokens $c_{i_j}, \ldots, c_{i_{j'-1}}$ do not appear in the tags of $\mathcal{T} \setminus \{t\}$, it follows that $(\mathcal{T} \setminus \{t\}) \cup \{t'\}$ is also optimal. Repeated application of this operation yields the lemma. □

Note that a periodic tag whose shortest period has length p contains as substrings exactly p c-tokens, while tags with the uniqueness property contain between $l - c + 1$ and $l - c/2 + 1$ c-tokens. Therefore, of the two classes of tags in Lemma 1, periodic tags (particularly those with short periods) make better use of the limited number of available c-tokens.

Each periodic tag corresponds to a directed cycle in the graph H_c which has $\mathcal{C}$ as its vertex set, and in which a token c_i is connected by an arc to token c_j iff c_i and c_j can appear consecutively in a tag, i.e., iff c_j is obtained from c_i by appending a single nucleotide and removing the maximal prefix that still leaves a valid c-token. Clearly, a vertex-disjoint packing of n cycles in H_c yields a feasible solution for MTSDP($l|C|multiple$) consisting of n tags, since we can extract at least one tag of length l from each cycle, and tags extracted from different cycles do not have common c-tokens. This motivates the following:

Maximum Vertex-Disjoint Directed Cycle Packing Problem: Given a directed graph G, find a maximum number of vertex-disjoint directed cycles in G.

The next theorem shows that Maximum Vertex-Disjoint Directed Cycle Packing in arbitrary graphs is unlikely to admit a polynomial approximation scheme.

Theorem 1. Maximum Vertex-Disjoint Directed Cycle Packing *is APX-hard even for regular directed graphs with in-degree and out-degree of 2.*

The proof of Theorem 1, which uses a reduction from the MAX-2-SAT-3 problem similar to the one in [6], can be found in [13]. A stronger inapproximability results was recently established for arbitrary graphs by Salavatipour and Verstraete [15], who proved that there is no $O(\log^{1-\varepsilon} n)$-approximation for Maximum Vertex-Disjoint Directed Cycle Packing unless $NP \subseteq DTIME(2^{polylogn})$. On the positive side, Salavatipour and Verstraete showed that Maximum Vertex-Disjoint Directed Cycle Packing can be approximated within a factor of $O(\sqrt{n})$ via linear programming techniques, matching the best approximation factor known for the edge-disjoint version of the problem [9].

Table 1. ILP results for MTSDP($l|C|1$), i.e., tag set design with specified tag length l, antitag-to-tag hybridization constraints, and a unique copy of each c-token allowed in a tag.

l	c	#tags		Upper Bounds		LP/ILP statistics				
		[12]	ILP	LP	[12]	#constr	#vars	#non-zero	LP time	ILP time
10	4	7	8	8.57	9	406	1878	6004	0.13	0.71
10	5	23	28	28.00	29	1008	4600	14596	2.27	5.85
10	6	67	85	85.60	96	2434	10940	34470	11.40	98.25
10	7	196	259	259.67	328	5808	25422	79274	86.70	586.67
10	8	655	–	853.33	1194	13554	57138	175492	552.74	–
20	4	3	3	3.53	3	926	4638	15244	1.05	58.46
20	5	9	10	10.50	11	2448	12240	40076	13.72	381.33
20	6	26	29	29.87	32	6354	31860	104270	182.96	12448.61
20	7	75	–	88.00	93	16528	82662	270194	2675.68	–
20	8	213	–	257.23	275	42834	213578	697292	134525.81	–

Table 2. ILP results for MTSDP($h|C|1$), i.e., tag set design with specified minimum tag weight h, antitag-to-tag hybridization constraints, and a unique copy of each c-token allowed in a tag.

h	c	#tags		Upper Bounds		LP/ILP statistics				
		[12]	ILP	LP	[2]	#constr	#vars	#non-zero	LP time	ILP time
15	4	6	7	7.00	7	610	2966	9612	0.45	9.04
15	5	18	21	21.09	21	1550	7456	23998	5.66	117.62
15	6	47	63	63.20	63	3830	18322	58752	54.43	2665.39
15	7	149	192	192.00	192	9406	44416	141638	544.95	3644.85
15	8	460	–	588.00	590	22766	105746	334904	7153.87	–
28	4	3	3	3.30	3	1286	6554	21624	1.88	132.78
28	5	8	9	9.67	9	3422	17388	57122	34.66	1137.21
28	6	22	27	27.48	27	8926	45518	149492	392.42	18987.09
28	7	64	–	78.55	78	23342	118828	389834	7711.41	–
28	8	175	–	–	224	60830	309118	1013244	–	–

We use a simple greedy algorithm to solve Maximum Vertex-Disjoint Directed Cycle Packing for the graph H_c: we enumerate possible tag periods in pseudo-lexicographic order, and check for each period if all c-tokens are available for the resulting tag. We refer to this algorithm as the *greedy cycle packing algorithm*, since it is equivalent to packing cycles greedily in order of length.

5 Experimental results

Tables 1 and 2 give ILP statistics (number of constraints, number of variables, and number of non-zero coefficients), LP and ILP runtime, and LP and ILP solution values for MTSDP($l|C|1$) and MTSDP($h|C|1$). We

Table 3. Results for MTSDP($*|C|multiple$), i.e., tag set design with antitag-to-tag hybridization constraints and multiple copies of a c-token allowed in a tag.

l/h	c	One c-token copy		Multiple c-token copies				
		Algorithm in [12]		Tree search		Cycle packing + Tree search		
		tags	c-tokens	tags	c-tokens	tags	c-tokens	% cyclic
$l = 20$	4	3	51	14	59	17	40	100.0
	5	9	146	31	165	40	140	100.0
	6	26	404	53	433	72	293	98.6
	7	75	1100	124	1179	178	928	99.4
	8	213	2976	281	3095	383	2411	97.1
	9	600	7931	711	8230	961	7102	96.9
	10	1667	20771	1835	21400	2344	19691	95.1
$h \geq 28$	4	3	58	14	61	17	40	100.0
	5	8	150	32	174	40	140	100.0
	6	22	398	44	432	72	300	98.6
	7	64	1119	118	1200	178	934	99.4
	8	175	2918	239	3037	379	2405	96.6
	9	531	8431	632	8622	943	6969	96.5
	10	1428	21707	1570	22145	2260	19270	94.1

also include the upper bounds established in [12] and [2] for these problems, and the number of tags found by using the alphabetic tree search algorithm in [12]. We solved all integer programs and their fractional relaxations using the CPLEX 9.0 commercial solver with default parameters run using a single CPU on a dual 2.8 GHz Dell PowerEdge 2600 Linux server. Missing entries did not complete in 10 hours.

The ILP solutions can be found in practical time for small values of c, which are appropriate for universal tag array applications, such as the emerging microfluidics-based labs-on-a-chip, where moderate multiplexing rates are sufficient and ensuring high hybridization stringency is costly. For all cases where the optimum could be computed, the difference between the optimal fractional and integer solution values was smaller than 1, indicating why CPLEX can solve to optimality these ILPs despite their size. Furthermore, ILP results confirm the extremely high quality of the upperbound established for MTSDP($h|C|1$) in [2]; the upperbound established in [12] for MTSDP($l|C|1$) appears to be somehow weaker.

Tables 3 and 4 give the results obtained for MTSDP($*|*|multiple$) by the alphabetic tree search algorithm described in Section 4, respectively by the greedy cycle packing algorithm (in our implementation, we impose an upper bound of 15 on the length of the cycles that we try to pack) followed by running the alphabetic tree search algorithm with the c-tokens occurring in the selected cycles already marked as unavailable. Performing cycle packing significantly improves the results compared to running the

Table 4. Results for MTSDP($*|\bar{C}|multiple$), i.e., tag set design with both antitag-to-tag and antitag-to-antitag hybridization constraints and multiple copies of a c-token allowed in a tag.

l/h	c	One c-token copy		Multiple c-token copies				
		Algorithm in [12]		Tree search		Cycle packing + Tree search		
		tags	c-tokens	tags	c-tokens	tags	c-tokens	% cyclic
$l = 20$	4	1	17	10	35	10	25	100.0
	5	4	65	17	83	23	85	100.0
	6	13	200	30	241	41	171	97.6
	7	37	537	68	585	97	512	99.0
	8	107	1480	147	1619	202	1268	98.0
	9	300	3939	362	4124	512	3799	96.3
	10	844	10411	934	10869	1204	10089	95.8
$h \geq 28$	4	1	22	10	36	10	25	100.0
	5	4	74	17	84	23	85	100.0
	6	12	213	29	238	41	178	97.6
	7	32	559	64	586	97	518	99.0
	8	90	1489	135	1632	199	1238	98.0
	9	263	4158	329	4314	504	3760	95.8
	10	714	10837	809	11250	1163	9937	93.6

alphabetic tree search algorithm alone; as shown in the tables, most of the resulting tags are found in the cycle packing phase of the combined algorithm.

Across all instances, the combined algorithm increases the number of tags by at least 40% compared to the MTSDP($*|*|1$) algorithm in [12]; the improvement is much higher for smaller values of c. Quite notably, although the number of tags is increased, the tag sets found by the combined algorithm use a *smaller* total number of c-tokens. Thus, these tag sets are less likely to cross-hybridize to the primers used in the reporter probes, enabling higher tag utilization rates during tag assignment [3, 12].

6 Conclusions

In this paper we proposed new solution methods for designing tag sets for universal DNA arrays. We have shown that optimal solutions can be found in practical time for moderate problem sizes by using integer linear programming, and that the use of periodic tags leads to increases of over 40% in the number of tags, with simultaneous increases in effective tag utilization rates during tag assignment. Our algorithms use simple greedy strategies, and can be easily modified to incorporate additional practical design constraints, such as preventing the formation of hairpin secondary structures, or disallowing specific nucleotide sequences such as runs of 4 identical nucleotides [11].

An interesting open problem is to find tight upper bounds and exact methods for the MTSDP($*|*|multiple$) formulations. Settling the approximation complexity of Maximum Vertex-Disjoint Directed Cycle Packing is another interesting problem.

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. A. Brenneman and A. Condon. Strand design for biomolecular computation. *Theor. Comput. Sci.*, 287(1):39–58, 2002.
5. S. Brenner. Methods for sorting polynucleotides using oligonucleotide tags. *US Patent 5,604,097*, 1997.
6. A. Caprara, A. Panconesi, and R. Rizzi. Packing cycles in undirected graphs. *Journal of Algorithms*, 48(1):239–256, 2003.
7. J.N. Hirschhorn et al. SBE-TAGS: An array-based method for efficient single-nucleotide polymorphism genotyping. *PNAS*, 97(22):12164–12169, 2000.
8. L. Kaderali. *Selecting Target Specific Probes for DNA Arrays*. PhD thesis, Köln University, 2001.
9. M. Krivelevich, Z. Nutov, and R. Yuster. Approximation algorithms for cycle packing problems. In *Proc. ACM-SIAM Annual Symposium on Discrete Algorithms*, pages 556–561, 2005.
10. M. Lothaire. *Combinatorics on Words*, volume 17 of *Encylopedia of Mathematics and Its Applications*, pages xix+238. Addison-Wesley, 1983.
11. 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.
12. I.I. Măndoiu, C. Prăjescu, and D. Trincă. Improved tag set design and multiplexing algorithms for universal arrays. In *Proc. 1st International Workshop on Bioinformatics Research and Applications (IWBRA)*, 2005 (to appear).
13. I.I. Măndoiu and D. Trincă. Exact and approximation algorithms for DNA tag set design. ACM Computing Research Repository, cs.DS/0503057, 2005.
14. 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.
15. M.R. Salavatipour and J. Verstraete. Disjoint cycles: Integrality gap, hardness, and approximation. In *Proc. 11th Conference on Integer Programming and Combinatorial Optimization (IPCO)*, 2005 (to appear).
16. J. SantaLucia. A unified view of polymer, dumbbell, and oligonucleotide DNA nearest-neighbor thermodynamics. *Proc. Natl. Acad. Sci. USA*, 95:1460–1465, 1998.
17. 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.