

A post-processing method for optimizing synthesis strategy for oligonucleotide microarrays

Kang Ning, Kwok Pui Choi¹, Hon Wai Leong and Louxin Zhang^{2,*}

Department of Computer Science, ¹Department of Statistics and Applied Probability and ²Department of Mathematics, National University of Singapore, Singapore 117543

Received May 26, 2005; Revised July 31, 2005; Accepted September 15, 2005

ABSTRACT

The broad applicability of gene expression profiling to genomic analyses has generated huge demand for mass production of microarrays and hence for improving the cost effectiveness of microarray fabrication. We developed a post-processing method for deriving a good synthesis strategy. In this paper, we assessed all the known efficient methods and our post-processing method for reducing the number of synthesis cycles for manufacturing a DNA-chip of a given set of oligos. Our experimental results on both simulated and 52 real datasets show that no single method consistently gives the best synthesis strategy, and post-processing an existing strategy is necessary as it often reduces the number of synthesis cycles further.

INTRODUCTION

Microarrays have been greatly successful for gene expression profiling in a variety of genomic analyses (1,2), such as SNPs detection, gene identification, drug discovery and clinical diagnosis. There are two major types of microarrays, namely, cDNA arrays and oligonucleotide arrays. While cDNA arrays are low cost, oligo arrays are designed to reduce cross-hybridization and hence to improve sensitivity. One leading technology for synthesizing oligo arrays is by a photolithographic method similar to that used in the semiconductor industry. In this method (2), the nucleotides A, C, G and T are added onto the appropriate spots on the array in a series of synthesis cycles. In each cycle, a specific mask is manufactured to permit light to penetrate only at specific positions for activating the oligonucleotides in the cells for synthesis. The pattern in which light penetrates the used masks directs the base-by-base synthesis of oligos on a solid surface.

The fabrication cost and time of an oligo array depends largely on the number of cycles required to synthesize the

given set of oligo probes. The fabrication of physical masks is a laborious and costly process. Even if a 'virtual masking' strategy can be used (3), the deprotection step for each cycle lasts ~ 5 min, and photolabile nucleosides are costly too. Therefore, it is important to manufacture an array of oligos in as few cycles as possible. Such a problem is called the synthesis strategy optimization problem. Another motivation for studying this problem is that it also reduces synthesis errors since masking is error-prone.

The simplest method for synthesizing a given set of oligos is to add A, C, G and T periodically. If the length of the given oligos is K , this method gives a strategy Φ of $4K$ cycles. However, the best synthesis strategy usually has many fewer than $4K$ cycles. Hubbell *et al.* (4) derived a better strategy from Φ by skipping a synthesis cycle if the nucleotide that is supposed to be added in the cycle is not needed by any oligos, or if the oligos that require this nucleotide can still be synthesized when it is deposited later. Since then, several methods for optimizing synthesis strategy have been proposed by many researchers (5–7). To derive a good synthesis strategy for constructing an array of oligos, Tolonen *et al.* (5) focused on both probes selection and the deposition order of nucleotides. Given a set of oligos, their deposition method is to add the most common nucleotide at the available bases of all the oligos in each step. Kasif *et al.* (6) developed a computational framework for a synthesis strategy, and proposed further greedy methods and their look-ahead extensions for potentially more efficient synthesis strategies.

As observed by Kasif *et al.* (6), Sven (7) and other researchers, optimizing the synthesis strategy for a given set of oligos is actually equivalent to the problem of finding a shortest common supersequence (SCS) for a set of strings, a well-studied algorithmic problem in computer science (8–12). Furthermore, the SCS problem can be seen as a special case of the multiple sequence alignment problems, any efficient method for multiple sequence alignment can also be used to optimize the synthesis strategy. However, these exact alignment-based algorithms are computationally too demanding, if not impossible, for synthesizing thousands of oligos in an array.

*To whom correspondence should be addressed. Tel: +65 6874 6579; Fax: +65 6779 5452; Email: matzlx@nus.edu.sg

In this paper, we assess a number of greedy methods and their look-ahead extensions on simulated and 52 real datasets. We have examined two approaches to reduce the number of synthesis cycles required to fabricate oligonucleotide arrays. The first approach focuses on choosing an efficient deposition order for nucleotides on an array. The second approach, which we call ‘post-processing’, streamlines a defined synthesis strategy by omitting unnecessary cycles. The post-processing approach, as shown in our results below, will often reduce the number of synthesis cycles further.

MATERIALS AND METHODS

Known methods for optimizing synthesis strategy

Suppose we are given a set of N K -mer oligos to be synthesized on a DNA-chip. A strategy for synthesizing them consists of a series of l cycles, where, in each cycle, a single nucleotide (A, C, G or T) is added to all unmasked partially constructed oligos, and at the end of the l -th cycle all the given N K -mers are deposited in their specified locations in the chip. Although the exact locations of these K -mers have an effect on the quality of the manufactured array, it is not an issue for optimizing the synthesis strategy as the strategy is only concerned about the ‘order’ of depositing the nucleotides.

Suppose we are to synthesize the given five 3mers as shown above the thick line in Figure 1a. Figure 1b–f illustrates the cycle-by-cycle progression in synthesizing these 3mers. After each cycle, we show the partially constructed oligos below the thick line and the remaining portions of the given 3mers above the thick black line. In each cycle, the nucleotide just added is underlined. The illustrated strategy requires 5 cycles and is optimal for the given 3mers, in the sense that there is no synthesis strategy that requires fewer than 5 cycles.

Assume the nucleotide added in the i -th cycle is $S[i]$ for each i from 1 to l . Then, we obtain a string $S = S[1]S[2] \dots S[l]$. Obviously, S is a common ‘supersequence’ of the given K -mers, i.e. each K -mer can be obtained by deleting some symbols from S . Therefore, optimizing the synthesis strategy is equivalent to finding an SCS of the given K -mers. This problem is well known to be NP-hard even for approximation (10).

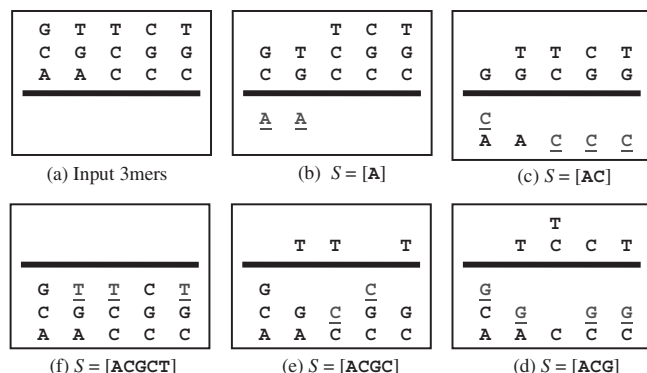


Figure 1. A cycle-by-cycle illustration of the synthesis process for five given 3mers: ACG, AGT, CCT, CGC and CGT. The partially constructed oligos are shown below the thick black line. (a) The five given 3mers. (b) The configuration after depositing nucleotide A (underlined). (c–f) The configurations after depositing nucleotides C, G, C and T, respectively. The synthesis strategy obtained is $S = [ACGCT]$.

Because of the NP-hardness, many researchers have proposed a variety of heuristic methods for finding good synthesis strategy. A simple ‘oblivious’ method is to add nucleotides A, C, G and T cyclically to the appropriate spots on the array until the entire array is synthesized. After every 4 cycles all partially constructed oligos will be one base longer; this simple method completes the synthesis in at most $4K$ cycles. From the approximation algorithm theory viewpoint, this algorithm has approximation ratio 4. A shortcoming of this algorithm is that it ignores the structures of the given K -mers, it always outputs a synthesis strategy consisting of $4K$ cycles even if an optimal strategy requires K cycles in some extreme cases. However, since the K -mer probes that have been pre-selected for constructing oligo arrays have special structure, a better masking strategy usually needs substantially fewer than $4K$ cycles (4–6).

Greedy methods

Two greedy methods were proposed in (6), which take into account of the structures of the given K -mers. To describe these methods, we adopt the notations used in (6). After t synthesis cycles, a partial oligo has been synthesized in each spot of the array. The height of each partially constructed oligo is defined as the number of bases in it. Indication of how much work that has been accomplished after t cycles (equivalently, how much more work needs to be done) is measured in terms of (i) the ‘minimum-height’—the height of the shortest partially constructed oligo after t cycles, or (ii) the ‘sum-height’—the sum of the heights of the partially constructed oligos. In the example shown in Figure 1e, the minimum height is 2 and the sum height is 12 ($= 3 + 2 + 2 + 3 + 2$).

The ‘minimum-height greedy method’, denoted here by MH and proposed in (6), selects a nucleotide that will extend the shortest partially constructed oligos by one base.

The ‘sum-height greedy’ method, denoted by SH and proposed in (5,6,8), selects a nucleotide that will result in the largest increase in the sum-height.

When there are more than one nucleotide which fulfill the height requirement at any cycle, we randomly pick one such nucleotide in these greedy methods.

Look-ahead extensions

A natural way to improve a greedy algorithm, say SH, is to apply a look-ahead strategy to it, which is commonly adopted in chess games (5,6). This strategy looks at a number of steps ahead before deciding which nucleotide best to be added. More specifically, choose two integers k and l such that $l \leq k$. The look-ahead extension of the SH method works as follows: (i) examine all possible partial oligos that can be generated in k cycles; (ii) for each such generated partial oligo, compute the height; and (iii) select the strategy (for the next l nucleotides) that will result in the largest increase of sum-height in k cycles. Here, we break the tie arbitrarily. This algorithm is called the (k, l) -look-ahead SH and is abbreviated to (k, l) -LA-SH. It is easy to see that an increase in k naturally leads to a better synthesis strategy, but also a substantial increase in computing time. The look-ahead extension of MH is defined in a similar fashion and is denoted by (k, l) -LA-MH. After some trials, it is found that (3, 1)-LA-SH gives the best trade-off of the two conflicting factors: synthesis

performance and computing time. We have tried (4, 1)-LA-SH and (5, 1)-LA-SH and observed not much an improvement over the (3, 1)-LA-SH but a huge increase in computing time. Hence in this paper, we shall consider (3, 1)-LA-SH.

We shall introduce a further refinement of the (k, l) -LA-SH. As the cycles are being executed using this synthesis strategy, the partially constructed oligos can be of very different heights. Switching this strategy, after a certain proportion of oligos have been fully deposited, to the MH algorithm may keep the heights of the remaining proportion of oligos to have roughly the same height. Conceivably, this may refine the (k, l) -LA-SH synthesis strategy. Here, we adopt the (3, 1)-LA-SH_m50 strategy, which applies (3, 1)-LA-SH until 50% of oligos have been fully deposited and then switches to the MH method. As it turns out (refer to the Supplementary Table), the results are mixed. In a few test datasets, there were some improvements in using this switching mechanism, while in others, the performance is slightly worse.

Throughout the rest of this paper, we shall simply refer to (3, 1)-LA-SH and its refinement, (3, 1)-LA-SH_m50, as LA and LA_m50, respectively.

The Hubbell–Morris–Winkler (HMW) method

This method appeared previously (4). The method consists of two key steps. First, it finds a shortest periodical strategy corresponding the following string: $(XYZU)(XYZU)\dots(XYZU)(XYZU)$, where X, Y, Z, U denote the four different nucleotides in some order. Second, examine each cycle from the first to the last and remove it if the nucleotide supposed to be added in the cycle is either not needed by any oligo or can be added in later cycles.

A new method

The HMW method can be generalized into the following template-based approach [see for example (13)]. Here, we propose a new method for post-processing a template strategy.

A template-based approach with look-ahead

We start with a synthesis strategy $S = S[1]S[2] \dots S[m]$, with m cycles. The following approach, using S as a template and a method AL, seeks to reduce the number of synthesis cycles without deteriorating the performance of the starting synthesis strategy S .

Input: a synthesis strategy $S = S[1]S[2] \dots S[m]$ of a set Π of K -mers, a method AL.

Iterate the following two steps until no further improvement is achieved:

- (i) For each position i from 1 to m
 - (a) For each K -mer $p = p[1]p[2] \dots p[K]$, Identify the longest suffix $p' = p[j_p]p[j_p + 1] \dots p[K]$ of p that occurs in $S_{\text{right}} = S[i + 1] S[i + 2] \dots S[m]$ as a subsequence, where $j_p = K - |p'| + 1$
 - (b) Apply the method AL to the oligos set $\{p[1] p[2] \dots p[j_p - 1] \mid p \in \Pi\}$ to get a strategy S' ;
 - (c) Break from Step 1 if S' is shorter than $S_{\text{left}} = S[1]S[2] \dots S[i]$.
 - (ii) Replace $S = S_{\text{left}}S_{\text{right}}$ with S'_{right} .
-

In this paper, we start with LA to obtain a template. Then, using LA, we apply the above post-processing approach to the template for further improvement. We coin this strategy as LAP [short for (3, 1)-Look Ahead with Post-processing]. We will also compare the performance of this strategy with other methods described in this paper.

RESULTS

Observations and simple analysis

Obviously, the HMW method outputs a strategy of at most $4K$ cycles on the input of a set of K -mer oligos. This is also true for the MH method. The MH method works greedily on the height of constructed oligos by extending the shortest partially constructed oligos by one base. After 4 cycles, all the shortest partially constructed oligos will be at least one base longer. Hence, the MH method always outputs a strategy of at most $4K$ cycles on a set of K -mer oligos.

Surprisingly, the SH method does not always output a strategy of at most $4K$ cycles on a set of K -mer oligos. Consider the following eleven 3mers:

aat, aaa, tat, cat, tta, att, gaa, ttt, ata, cca, tct.

One optimal strategy of this oligo set is [ccgatatat] with 9 cycles since any strategy requires three cycles each to print nucleotides a and t, two cycles and one cycle for c and g, respectively. But the SH method outputs a strategy such as [tatatcatgaaca], which requires 13 ($>3 \times 4$) cycles. This fact is also observed from the simulation test results given in Table 1.

On the other hand, the SH method can give better strategy than the MH method. Consider the following example with sixteen 16mer oligos:

```
aaaaggggcccctttt, cccctttttaaagggg,
aaaggggcccctttta, ccctttttaaagggggc,
aaggggccccttttaa, cctttttaaagggggc,
agggggcccctttttaa, ctttttaaagggggccc,
ggggcccctttttaa, tttttaaagggggccc,
gggcccctttttaaag, ttttaaagggggcccct,
ggcccctttttaaag, ttaaagggggcccctt,
gccctttttaaaggg, taaagggggccccttt,
```

where positions of bases in the oligos are listed from left to right. The following optimal strategy S^* requires 31 cycles:

$S^* = [aaaagggggcccctttttaaagggggccccttt]$.

However, in the worst case, the MH method outputs a strategy that requires 4 cycles to add nucleotides each position except for positions 5, 9, 13, 16 and requests 3 cycles for each of the positions 5, 9, 13, 16. This results in 60 cycles. Therefore, the worst-case approximation ratio of the MH method is at least 60/31. On the same set of 16mers, the SH method outputs a strategy that requires 58 cycles in the worst case. And so, the worst-case approximation ratio of the SH method is at least 58/31. We omit the details due to space constraint. It seems possible that one could prove that both SH and MH methods do not have approximation ratio better than 2 in worst case.

Table 1. A baseline comparison of the performance of different methods (SH, MH, LA, LA_m50, HMW and LAP) over different oligo lengths (K) and number of oligos (N)

K	N	SH	MH	LA	HMW	LAP
25	10 000	87.7 ± 4.2	82.7 ± 1.5	82.5 ± 1.5	81.9 ± 0.9	81.7 ± 1.6
	20 000	88.9 ± 4.8	83.1 ± 1.2	83.4 ± 1.5	82.8 ± 0.8	82.0 ± 1.5
	40 000	88.9 ± 4.6	83.4 ± 1.1	83.6 ± 1.4	83.1 ± 0.9	82.3 ± 0.9
50	10 000	166.6 ± 12.9	151.7 ± 4.3	150.3 ± 3.7	150.5 ± 2.8	147.7 ± 3.7
	20 000	167.3 ± 13.2	151.5 ± 4.3	150.9 ± 3.5	151.0 ± 2.9	148.9 ± 3.5
	40 000	168.0 ± 13.2	152.1 ± 4.4	151.0 ± 3.5	151.1 ± 3.0	148.7 ± 3.4
100	10 000	302.5 ± 16.3	286.2 ± 10.2	281.9 ± 10.0	288.4 ± 5.0	279.9 ± 10.0
	20 000	302.4 ± 16.1	285.9 ± 10.1	281.5 ± 10.3	288.8 ± 5.1	279.4 ± 10.3
	40 000	303.2 ± 16.2	286.5 ± 10.2	281.9 ± 10.7	289.1 ± 5.8	279.7 ± 10.4
200	10 000	570.8 ± 21.3	549.2 ± 22.0	540.5 ± 23.8	560.2 ± 11.8	537.2 ± 23.7
	20 000	571.0 ± 20.3	549.8 ± 21.1	540.8 ± 23.5	560.5 ± 11.2	538.2 ± 23.4
	40 000	572.0 ± 20.3	550.6 ± 21.1	542.0 ± 23.1	558.3 ± 10.6	538.3 ± 23.3

Each entry gives the average and standard deviation of the number of the synthesis cycles in the strategy output from the corresponding method. An entry in bold indicates that the corresponding method is the best for that (K , N) combination.

Assessing the greedy algorithms and their extensions

To assess the performance of different algorithms, we implemented the SH, MH, HMW, LA, LA_m50 and LAP methods, and tested them on both random datasets and all the available 52 real datasets. For randomly generated datasets, we assessed these methods in terms of the mean and the standard deviation of the number of synthesis cycles in the synthesis strategy produced from 20 simulated datasets, each characterized by the parameters (K , N , p) where K is the length of each oligo, N is the number of K -mers, and p is the GC content of the dataset. We use the GC content, p , as a problem parameter for two reasons: (i) if the GC content is low (<20%), the oligos are composed mostly of A's and T's, and therefore the oligos are more 'similar', and we expected to have shorter synthesis strategies; and (ii) GC are rich in gene coding sequences and GC content relates to melting temperatures, real datasets often have GC contents in the range 40–50%, and consequently, we have included randomly generated datasets with GC contents in that range. Symmetry consideration shows that we only need to study $p \leq 50\%$. These datasets were generated using our program 'Random DNA Oligo Generator' (available upon request).

Baseline performance comparison on simulated datasets

For our first baseline performance comparison, we assess all the methods except for LA_m50 using randomly generated datasets over a broad range of parameters, with $K = 25, 50, 100, 200$, and $N = 10\,000, 20\,000$ and $40\,000$. For each combination of (K , N), we generate 100 datasets. Although the real datasets often contain 25mer oligos, 200mer probes are reported to be the limit for the phosphoramidite approach. Hence, we also evaluate these methods on long oligo datasets for potential applications in the future. For $K = 25$, there are 4^{25} ($\approx 1.1 \times 10^{15}$) different 25mer oligos. For $N = 10\,000, 20\,000$ and $40\,000$, the datasets contains up to $\sim 4 \times 10^5 / 10^{15}$ of all the possible 25mers.

Table 1 summarizes the performance comparison of the different methods on these baseline datasets. Each row represents the average standard deviation for each (K , N) combination of over 100 datasets with different GC contents. An entry in bold indicates that the corresponding method is the best for that (K , N) combination. Overall, the look-ahead methods

show significant improvements over the greedy methods SH and MH. The difference is greater for longer oligos, e.g. when $K = 200$ and $N = 40\,000$, they require on average ~ 20 fewer cycles than MH and ~ 40 fewer cycles than SH. LAP is the best overall and it further reduces the number of cycles required by the LA method by ~ 2 in most datasets.

In terms of running time, post-processing algorithms LAP and HMW are much slower than the greedy methods. When $K = 200$ and $N = 40\,000$, the greedy methods required 1–2 min to process an oligos dataset, while the LAP and HMW took more than an hour.

Effect of GC content

In Table 1, the length of the synthesis strategy for (K , N) combination appears to have a rather high standard deviation. This is true for all the methods studied. To examine the effect of sequence variation on the length of the optimal strategy, we use the GC content as a proxy measure of the variation of the oligos in a dataset. Hence, we segregate the datasets by GC content (p) to get the table in the Supplementary Data. An entry in bold indicates that the corresponding method is the best for that (K , N , p) combination. (To keep the table small, we only show $p = 20, 40$ and 50% .) For each method, the average length of the synthesis strategy varies for different GC contents, p . However, for a given value of p , the small standard deviation indicates that the length of the synthesis strategy does not differ much among the (K , N , p) datasets. We note that the effect of GC content on the length of the synthesis strategy is observed for all the methods studied.

For datasets where $K = 25$, $N = 10\,000, 20\,000, 40\,000$, and $p = 20, 40$ and 50% , we noted that the LA generally output shorter synthesis cycles than the SH and MH did when GC content is in the range from 20 to 40%. The MH method also performs better than SH when GC content is in the range from 20 to 40%, but slightly worse than the SH when GC content is 50%. The performance of SH and HMW methods improves significantly as the GC content increases to 50%. This seems to suggest that the SH and HMW methods are sensitive to the GC content of the oligos.

We also noticed that generally LAP method performed better than the HMW method for the datasets when GC content is low (say, 20%) and the difference is greater with longer oligos: ~ 38 cycles for $K = 200, N = 40\,000$ and $p = 20\%$. However,

the performance of HMW improves when GC content increases to ~40–50% and it becomes comparable with the LAP, but the LAP is still slightly better as indicated by the number of bold entries in the table. For example, when $K = 25$, $N = 40\,000$ and $p = 40\%$, the LAP method outputs shorter synthesis cycles in 8 out of 20 datasets; whereas HMW method in 2 out of 20 datasets. When $K = 25$, $N = 40\,000$ and $p = 50\%$, the LAP method outputs a shorter synthesis cycles in 8 out of the 20 datasets, while the HMW method in 5 out of the 20 datasets.

Next, we assess the performance on data with high GC content, namely, with p between 40 and 50% (as is common in the available real datasets). Specifically, we consider 360 datasets where $K = 25$, $N = 10\,000$, $20\,000$, $40\,000$, and $p = 40, 42, 44, 46, 48, 50\%$. For these datasets, we count the number of times a particular method outputs the best strategy on the 360 datasets. When a particular method outputs the best strategy, it will receive one vote. When more than one method tied for the best strategy, each of these strategies receives one vote. The SH method receives 157 votes, LA 153 votes, MH 108 votes and LA_m50 89 votes. The result shows that the LA and SH methods performed best more often for the random datasets when GC content ranges from 40 to 50%; this is consistent with the results given in the table.

Performance on the real oligo datasets

Fifty-two photolithographic *in situ* synthetic array datasets (<http://www.affymetrix.com/support/technical/byproduct.affx?cat=arrays>) were also used to test the methods. The number of 25mer oligos in all the available 52 real datasets ranges from 6000 to 675 000. In addition to testing the methods on these real oligo-chip datasets, we also computed the lower bound on the optimal number of synthesis cycles, as an indicator of how well the methods perform. For each dataset, we first selected four oligos that contain the largest number of nucleotides A, G, C, T respectively. Then, we apply dynamic programming to find a best synthesis strategy for these four oligos. The lower bounds obtained for the real datasets are in the range from 52 to 69, the most of which are from 60 to 63.

Our experimental results show that the LA approach outperformed significantly the greedy methods SH and MH in two aspects: (i) in 47 out of 52 cases, it outperformed the SH and MH methods; and (ii) in 34 out of 52 cases, the synthesis strategy output from the LA approach is at least 10 cycles shorter than the ones from SH and MH. In fact, on all the datasets except for *C_elegans*, *P_Anopheles*, the LA and its variation LA_m50 have similar performance. In each case, they output a synthesis strategy with 70–73 cycles. On the other hand, the MH never outperformed LA's; when the SH method outperformed LA in 5 out of 52 real datasets, the difference is marginal with only 1 or 2 cycles shorter.

In 40 out of 52 cases, the LAP shaved off the strategy output from the LA further by 1 or 2 cycles. Hence, we conclude that such a post-processing procedure is efficient. Running on a machine with 2.0 GHz CPU and 1G memory, the LAP took ~1–20 min to complete, depending on the size of the dataset. In the worst case, it took ~40 times as much computational time as the LA method did. Since this additional computation

is only a one-time cost per set of oligos, the benefits from reducing possible errors due to masking and the savings from time and money in manufacturing large quantities of the same array far outweigh the extra computation time.

Good synthesis strategy for the 52 real oligo-array datasets

The synthesis strategies output from our template-based post-processing program on all the real datasets are also examined. These good synthesis strategies have not appeared in literature and could be valuable to DNA-chip manufacturers. To get these good strategies, we ran both the HMW and our LAP methods on these 52 datasets. Both methods output (in most cases, different) strategies with 70 synthesis cycles on all the datasets except for *C_elegans*, *P_Anopheles*, Soybean and Test3; one of such strategies is $(TGCA)^{17}TG$, where we use superscript 17 to denote the concatenation of 17 copies of TGCA. For *C_elegans* and *P_Anopheles* datasets, the HMW method outputs a strategy [e.g. $(CATG)^{17}CA$] of 70 synthesis cycles, but our LAP method outputs a strategy of >80 cycles. For the Test3 dataset, both methods output a strategy of 72 cycles, one of which is $(TGCA)^{18}$. Interestingly, for the Soybean dataset, our method outputs a strategy [e.g. $A(TGCA)^{17}TGT$] of 72 cycles, one cycle shorter than that of the HMW method. Hence, we obtained a better strategy for the Soybean chip dataset.

DISCUSSION

Wide ranging applications of gene expression profiling have generated a great demand of DNA-chips. Improving the cost effectiveness in fabricating these chips will promote an even broader applicability of microarray technology. This work focuses on optimization methods for finding synthesis strategy in DNA-chip manufacture. The greedy methods SH and MH were proposed independently by many researchers for tackling this problem or its equivalent problem—the SCSs. The look-ahead approach is a natural extension of the greedy algorithms proposed in the work (6) without extensive comparison. Here, we compared the performance of the SH and MH methods, their look-ahead extensions, the HMW method and the LAP method on both simulated and real datasets.

Our experimental results show that the look-ahead approach outperforms significantly the greedy algorithms SH and MH in simulated datasets especially on longer oligos or when GC content is lower (or higher). Our results on simulated dataset also show clearly that the SH performs better, while the MH performs poorer, when GC content increases. For most of the real datasets, the look-ahead approach outputs a synthesis strategy with 70–73 cycles, ~10 cycles shorter than the greedy algorithms.

We also propose a template-based method that attempts to obtain better strategy by post-processing an existing strategy. When such a method is applied to the output from the LA method on a real dataset, we usually obtain further improvement. The program code is given in the Supplementary Data. The overall superb performance of the HMW and LAP methods on both simulated and real datasets indicates that

the post-processing approach is very effective for obtaining good synthesis strategy.

On simulated datasets with 25mers of GC content in the range from 40 to 50%, all the methods performed similarly. Each of them outputs strategy of ~ 80 cycles. However, on the real datasets with similar GC content, the LAP and HMW methods output strategy with ~ 70 cycles, at least 10 cycles shorter than the strategies given by the greedy methods. This demonstrates that how to best select regions of each gene to be used for oligos is also extremely important for optimizing synthesis strategy. The reader is referred to (5,14,15) for the study in this aspect. Hence, taking both oligo selection and nucleotide deposition order into account (5) seems to provide a promising approach to optimizing synthesis strategy.

In addition, we provide the best strategies obtained for all the 52 real datasets. Such a list is valuable to DNA-chip manufacturers. It can also be used as a benchmark for assessing new methods for optimizing synthesis strategy.

Finally, microarray fabrication motivates many interesting algorithmic problems in selection and deposition of oligos (16). For example, as more and more genes are known in a specific genome, one DNA-chip may no longer be large enough to host the set of selected oligos; therefore, multiple DNA-chips for one genome are needed in the future. An important question on how best to distribute these oligos into different chips effectively so that the maximum (or total) number of cycles of these strategies for these chips is minimized will be of practical and theoretical interest. Our hope is that the study here contributes to some good starting points for attacking this general problem in the future.

SUPPLEMENTARY DATA

Supplementary Data are available at NAR Online.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous reviewers for many helpful comments and suggestions which improve this work, including the suggestion to experiment with longer oligos. They also thank Pavel Pevzner for discussion on the algorithms, Andy C. Tolonen for useful email discussion on the topic and Kok Seng Chua for help in the beginning of this project. The work of K.P.C. and L.Z. was partially supported by Singapore BMRC research grant BMRC01/1/21/19/140

and a NUS ARF research grant and the work of H.W.L. was partially supported by the National University of Singapore under grant R252-000-199-112. Funding to pay the Open Access publication charges for this article was provided by Singapore BMRC.

Conflict of interest statement. None declared.

REFERENCES

1. Schena, M., Shalon, D., Davis, R.W. and Brown, P.O. (1995) Quantitative monitoring of gene expression patterns with a complementary DNA microarray. *Science*, **270**, 467–470.
2. Chee, M., Yang, R., Hubbell, E., Berno, A., Huang, X.C., Stern, D., Winkler, J., Lockhart, D.J., Morris, M.S. and Fodor, S.P. (1996) Accessing genetic information with high-density DNA arrays. *Science*, **274**, 610–614.
3. Singh-Gasson, S., Green, R.D., Yue, Y., Nelson, C., Blattner, F., Sussman, M.R. and Cerrina, F. (1999) Maskless fabrication of light-directed oligonucleotide microarrays using a digital micromirror array. *Nat. Biotechnol.*, **17**, 974–978.
4. Hubbell, E.A., Morris, M.S. and Winkler, J.L. (1996) Computer-aided engineering system for design of sequence arrays and lithographic masks. US Patent no. 5571639.
5. Tolonen, A.C., Albeau, D.F., Corbett, J.F., Handley, H., Henson, C. and Malik, P. (2002) Optimized *in situ* construction of oligomers on an array surface. *Nucleic Acids Res.*, **30**, e107.
6. Kasif, S., Weng, Z., Derti, A., Beigel, R. and DeLisi, C. (2002) A computational framework for optimal masking in the synthesis of oligonucleotide microarrays. *Nucleic Acids Res.*, **30**, e106.
7. Sven, R. (2003) Fast large scale oligonucleotide selection using the longest common factor approach. *J. Bioinform. Comput. Biol.*, **1**, 343–361.
8. Foulser, D., Li, M. and Yang, Q. (1992) Theory and algorithms for plan merging. *Artif. Intell.*, **57**, 143–181.
9. Bonizzoni, P. and Eherenfeucht, A. (1995) Approximation of the Shortest Common Supersequence with ratio less than the alphabet size. *Technical Report 149–195, Dipartimento di Scienze dell' Informazione*.
10. Jiang, T. and Li, M. (1997) On the approximation of shortest common supersequences and longest common subsequences. *SIAM J. Comput.*, **24**, 1122–1139.
11. Bonizzoni, P., D'Alessandro, M., Della Vedova, D. and Mauri, G. (1998) Experimenting and Approximation algorithm for the LCS. *Proceedings of the International workshop on Algorithms and Experiments*, 96–102.
12. Gaia, N. and Gianpaolo, O. (2003) An approximate A^* algorithm and its application to the SCS problem. *Theor. Comput. Sci.*, **290**, 2021–2029.
13. Sven, R. (2003) The shortest common supersequence problem in a microarray production setting. *Bioinformatics*, **19** (Suppl. 12), 156–161.
14. Li, F. and Stormo, G. (2001) Selection of optimal DNA oligos for gene expression analysis. *Bioinformatics*, **17**, 1067–1076.
15. Rouillard, J.M., Zuker, M. and Gulari, E. (2003) OligoArray 2.0: design of oligonucleotide probes for DNA microarrays using a thermodynamic approach. *Nucleic Acids Res.*, **31**, 3057–3062.
16. Hannenhalli, S., Hubbell, E., Lipshutz, R. and Pevzner, P.A. (2002) Combinatorial algorithms for design of DNA arrays. *Adv. Biochem. Eng. Biotechnol.*, **77**, 1–19.