Barcode Selection and Layout Optimization in Spatial Transcriptomics

Frederik L. Jatzkowski
Martin Luther University Halle-Wittenberg, Germany

Antonia Schmidt
Martin Luther University Halle-Wittenberg, Germany

Robert Mank
Martin Luther University Halle-Wittenberg, Germany

Steffen Schüler
Martin Luther University Halle-Wittenberg, Germany

Matthias Müller-Hannemann
Martin Luther University Halle-Wittenberg, Germany

Abstract

An important special case of the quadratic assignment problem arises in the synthesis of DNA microarrays for high-resolution spatial transcriptomics. The task is to select a suitable subset from a set of barcodes, i. e. short DNA strings that serve as unique identifiers, and to assign the selected barcodes to positions on a two-dimensional array in such a way that a position-dependent cost function is minimized. A typical microarray with dimensions of 768×1024 requires 786,432 many barcodes to be placed, leading to very challenging large-scale combinatorial optimization problems.

The general quadratic assignment problem is well-known for its hardness, both in theory and in practice. It turns out that this also holds for the special case of the barcode layout problem. We show that the problem is even hard to approximate: It is MaxSNP-hard. An ILP formulation theoretically allows the computation of optimal results, but it is only applicable for tiny instances. Therefore, we have developed layout constructing and improving heuristics with the aim of computing near-optimal solutions for instances of realistic size. These include a sorting-based algorithm, a greedy algorithm, 2-OPT-based local search and a genetic algorithm. To assess the quality of the results, we compare the generated solutions with the expected cost of a random layout and with lower bounds. A combination of the greedy algorithm and 2-OPT local search produces the most promising results in terms of both quality and runtime. Solutions to large-scale instances with arrays of dimension 768×1024 show a 37% reduction in cost over a random solution and can be computed in about 3 minutes. Since the universe of suitable barcodes is much larger than the number of barcodes needed, this can be exploited. Experiments with different surpluses of barcodes show that a significant improvement in layout quality can be achieved at the cost of a reasonable increase in runtime. Another interesting finding is that the restriction of the barcode design space by biochemical constraints is actually beneficial for the overall layout cost.

2012 ACM Subject Classification Applied computing → Operations research; Theory of computation → Mathematical optimization

Keywords and phrases Spatial Transcriptomics, Array Layout, Optimization, Computational Complexity, GPU Computing, Integer Linear Programming, Metaheuristics

Digital Object Identifier 10.4230/LIPIcs.SEA.2024.17

1 Introduction

We study a challenging large-scale combinatorial optimization problem that arises in the synthesis of microarrays for high-resolution spatial transcriptomics [26, 41], a rapidly evolving molecular profiling method that allows scientists to measure the gene activity in a tissue

sample and map where that activity occurs. At a more abstract level, we consider a special case of the famous quadratic (semi-)assignment problem. The task is to select a suitable subset from a given universe of barcodes and to assign the selected barcodes to the positions (sites) of an array in such a way that a position-dependent cost function is minimized.

Motivation and background. DNA barcodes are short strings of fixed length over the alphabet $\{A, C, G, T\}$. In bioinformatics, they are commonly used as tags in pooled sequencing experiments to enable the identification of reads originating from the same cell. Applications include the study of gene expression at the single-cell level [17], lineage tracing and screening [16], the exploration of developmental trajectories, progression and anti-tumor drug discovery for cancer therapy [2, 40], DNA data storage [1], and high-resolution spatial transcriptomics [26, 41]. The purpose of a barcode is to act as a unique identifier. This means that barcodes must be as easily distinguishable from each other as possible. They must be robust in experimental environments where errors alter the original barcodes by unintentional substitutions, insertions and deletions. Such errors can occur at all stages of the experimental workflow. They can be introduced during barcode synthesis, during the course of the experiment and in the final sequencing [33]. Modern techniques such as photolithographic microarray synthesis generate barcodes on the array in multiple rounds using computer-controlled micromirrors which nowadays replace physical masks. In each round, a subset of all sites (array positions) is exposed to light to activate oligonucleotides for further synthesis. In round i, a particular nucleotide $s_i \in \{A, C, G, T\}$ is available to be added to the light-exposed sites. Typically, the nucleotide deposition sequence $S = s_1, s_2, \ldots, s_K$ is assumed to be periodic, i.e. $S = (ACGT)^k$ for a sufficiently large k such that all barcodes can be generated [15]. Barcode synthesis faces serious manufacturing challenges due to unintended illumination effects such as scattered or diffracted light which cause a significant amount of error [21]. Experimental analysis has shown that photolithographic microarray synthesis produces barcodes with a comparatively high nucleotide error rate in the range of 10-20% per base [25]. Figure 1 visualizes the principle of the microarray synthesis and how errors depending on adjacent barcodes occur during this process.

This motivates the combinatorial optimization problem studied in this paper. For a given set of barcodes, we try to optimize the layout of the barcodes on the given array by minimizing the dissimilarity of neighboring barcodes. The main goal of this work is to achieve a significant reduction in barcode synthesis errors through a clever combination of barcode selection and layout optimization.

Barcode set construction. The construction of large barcode sets is a problem in itself which has been studied intensively, but is not the focus of this paper. The obvious minimum requirement for barcode sets is that all barcodes are unique. However, they should also be designed so that errors can be corrected. The Levenshtein (or edit distance) of two strings of length ℓ is the minimum number of insertions, deletions, and substitutions required to transform one string into the other [23]. A Levenshtein distance of at least d between any two barcodes provides the worst-case guarantee that up to $(d-1)/2$ errors can be corrected. Since the length of corrupted barcodes can vary, it is even preferable to use a generalized editing metric, the so-called sequence Levenshtein (SL) distance [3].

In addition, for biochemical reasons barcodes should satisfy several additional constraints such as a narrow GC-content range between 40 and 60%, avoidance of homopolymers and repeats of length ≥ 3, i.e. substrings of the type $(X)^k, (XY)^k, (XYZ)^k, \ldots$ for $k \geq 3$, and a distance from special sequences (primers, promoters, flowcell attachments) [6, 7, 38]. The

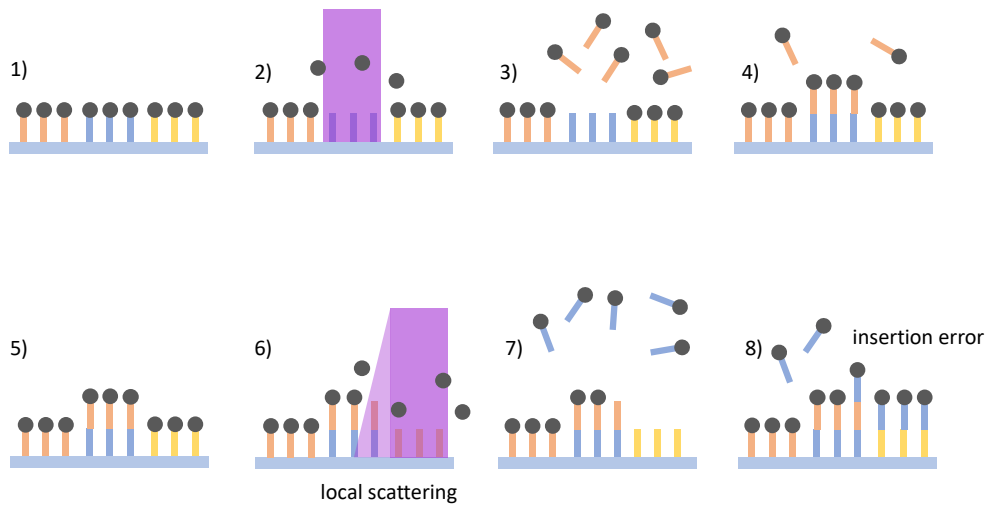


Figure 1 Microarray synthesis using photolithography, insertion error due to local scattering. The microarray appears as a blue bar. The barcodes are shown as colored sticks attached to the microarray, partially with protective layers displayed as dark gray circles. 1) Protective layer on all barcode stubs. 2) UV-exposure removes protective cover. 3) Microarray is flooded with nucleotides. 4) Nucleotides bind with unprotected barcodes stubs. 5) Excessive nucleotides are removed, next synthesis cycle begins. 6-8) Local scattering uncovers neighboring barcode stub, leading to an insertion error.

construction of large barcode sets can be done by rejection sampling. The crucial observation is that if we generate barcode candidates of sufficient length, say of length 30-40, as uniformly random DNA strings, then any two barcodes will have a relatively large Levenshtein distance with high probability [33]. Hence, one can easily generate as many barcode candidates as needed. In the sampling scheme, barcodes that do not fulfill some of the side constraints can be rejected. We will exploit the possibility to choose from large barcode sets to improve the cost of the layout.

Contribution. In this paper, we present the following results.

- By giving an L-reduction from `path-TSP` on Hamming spaces we show that the barcode layout problem is MaxSNP-hard and thus also APX-hard. Therefore, it does not admit a polynomial-time approximation scheme (PTAS) unless P=NP.
- We study and compare several lower bounds. These include the LP relaxation of an integer linear programming (ILP) formulation and three combinatorial lower bounds, including an adaptation of the Gilmore-Lawler bound for the QAP [9, 22], a simple combinatorial bound proposed by Kahng et al. [15] and a b-matching relaxation.
- To solve the barcode layout problem in practice, we consider several approaches: different versions of greedy-type algorithms, a 2-OPT-based local search, and a genetic approach. To achieve reasonable running times for solving large instances with these heuristics, most of them had to be implemented on a GPU.
- We present a computational study on a large array with dimensions 768×1024. In experiments, we investigate instances with the same number of barcodes as positions to be filled. To assess the quality of the results, we compare the generated solutions with the expected cost of a random layout and with lower bounds. Interestingly, our

GPU implementation of a greedy algorithm produces promising results in terms of both quality and runtime. Solutions to large instances with 768×1024 many positions show a 37% reduction in cost over a random solution and can be computed in about 94 seconds. Marginal additional improvements can be achieved by performing a local search starting from the greedy solution at moderate additional computational cost.

- Other experiments examine the effect of being able to select barcodes from larger sets with varying degrees of redundancy. We observe that the layout costs can be significantly improved, with a nice trade-off between quality improvement and additional runtime.
- In a final experiment, we explore the problem under different side constraints that barcodes have to fulfill. Very interestingly, adding extra constraints on the barcode type can be beneficial to the overall cost of the layout.

Related work. The placement of probes arises in a similar way in the design of DNA, protein, and peptide microarrays. It was first discussed under the name border length minimization problem (BLMP) by Hannenhalli et al. [11]. At that time, the standard fabrication technology used a sequence of masks instead of micromirrors, but the basic principle of synthesis in rounds was the same. Each mask exposes a subset of sites to light, namely exactly those where a nucleotide is to be coupled in the current round. Optical effects such as diffraction or reflection of light can cause unwanted illumination and therefore activation of sites adjacent to those that are intentionally exposed to light. To reduce this risk, one seeks a placement of probes minimizing the sum of border lengths in all masks. The NP-hardness of the BLMP has been shown in [32] (but allowing unbounded alphabets for strings as input, while our alphabet is fixed to size four) and for rectangular and square arrays in [20, 21]. Kahng et al. [15] distinguish between synchronous and asynchronous DNA array synthesis. In synchronous synthesis, the i-th period ($ACGT$) of the periodic schedule synthesizes a single nucleotide in each probe, whereas asynchronous array synthesis permits any number between 1 and 4 of nucleotides in any given period, allowing shorter synthesis schedules. Finding an optimal deposition sequence (often referred to as *embedding*) of probes adds another challenging dimension to the BLMP. The combined optimization of placement and embedding has been studied by [5]. For simplicity and to minimize the number of synthesis cycles, in this paper we always use a fixed deposition schedule with a leftmost embedding strategy.

A few approximation results are known for the BLMP. Li et al. [24] show that the special case of a one-dimensional array (i.e. a single row) can be approximated by a constant factor. For asynchronous synthesis, they show that the BLMP is $\sqrt{n}\log^2 n$-approximable when n probes need to be placed. This approximation guarantee has been improved to a factor of $n^{1/4}\log^2 n$ by Popa et al. [32]. Simple combinatorial lower bounds have been proposed by Kahng et al. [15]. The main difference of the classical placement problem to our version is that the given set of probes is usually fixed, whereas the barcode layout problem has the freedom to select from a much larger set of barcodes. The classical BLMP considers the 4-neighborhood of each site as a proxy for measuring the risk of unintended light exposure. The 8-neighborhood of each site includes not only adjacent sites along edges but also at corners. A natural extension is to apply more general distance functions [4], for example using all sites up to a distance of two or three with an appropriate weighting of the distance.

Carvalho and Rahmann [4] recognized the microarray layout problem as a special case of the quadratic assignment problem (QAP). The barcode layout problem studied in this paper can be formulated in a similar way. The QAP is one of the cornerstone problems in combinatorial optimization with a long and rich research history, see for example the surveys [8, 27, 36]. The QAP has a wide range of applications in facility location, scheduling,

b1: A C T G C A G T

b2: T A G T A C T G

	Cycle 1				Cycle 2				Cycle 3				Cycle 4			
	A	C	G	T	A	C	G	T	A	C	G	T	A	C	G	T
b1	1	1	0	1	0	0	1	0	0	1	0	0	1	0	1	1
b2	0	0	0	1	1	0	1	1	1	1	0	1	0	0	1	0

Figure 2 Example: Two barcodes of length $\ell = 8$ and synthesis distance $d(b_1, b_2) = 8$.

transportation, and placement problems in VLSI design. It was originally introduced by Koopmans and Beckmann in 1957 as a mathematical model for allocating a set of n facilities to n locations [18]. The QAP is well-known for its theoretical and practical intractability. It includes hard combinatorial optimization problems such as the traveling salesman problem (TSP), graph partitioning and maximum clique as special cases. Exactly solving instances with $n > 30$ is typically infeasible in practice, that is, such instances cannot be solved in reasonable computational time. Sahni and Gonzalez [34] showed that the QAP is NP-hard and even finding an ϵ-approximate solution is hard. For the general QAP it is even hard to find locally optimal solutions, it is known to be PLS-complete for a Kernighan-Lin type neighborhood and the 2-OPT (or pair exchange neighborhood) [31, 35]. Due to its difficulty, all kinds of meta-heuristics have been applied to QAP [19]. However, their performance depends strongly on the specific application. These meta-heuristics can be combined with a hierarchical refinement algorithm [21].

Overview. The rest of this paper is organized as follows. In Section 2, we formally introduce the barcode layout optimization problem. Then, in Section 3 we study its computational complexity and show its MaxSNP-hardness. Section 4 discusses several lower bounds. As the problem is computationally hard we consider several heuristic approaches in Section 5. In Section 6, we present a computational study evaluating these heuristics on large instances. Finally, in Section 7, we conclude with a brief summary. Source code and data are made freely available at `https://github.com/uni-halle/BarLay`.

2 Problem Definition

In order to provide a formal problem definition, we first have to formalize the term barcode. For our purposes, a barcode is a word of fixed length ℓ over the alphabet $\{A, C, G, T\}$. We define a barcode layout as a function that assigns barcodes to array positions.

▶ **Definition 1** (barcode layout). *Let $w, h \in \mathbb{N}$ be dimensions of a two-dimensional array and B with $|B| \geq w \cdot h$ a set of barcodes of length ℓ. The injective layout function $L(x, y)$ assigns a barcode $b \in B$ to each position (x, y) with $1 \leq x \leq w$ and $1 \leq y \leq h$.*

During photolithographic synthesis of a microarray, each barcode b of length ℓ will be uniquely associated with its synthesis schedule $s(b)$, which is a bitstring of length $\leq 4\ell$. This schedule determines in which synthesis cycles the barcodes should grow on the microarray. Figure 2 shows an example of barcodes and their associated synthesis schedules. Insertion errors can occur if a barcode is growing in a synthesis cycle it is not supposed to. Typically this happens when a neighboring barcode is illuminated and the light scatters. In this case, the synthesis schedules of the two barcodes are different for the cycle in question. Consequently, the risk of unwanted insertions can be quantified by the Hamming distance of the synthesis schedules. This leads to a natural distance measure between synthesis schedules.

▶ **Definition 2** (synthesis distance of barcodes). *Let b_1 and b_2 be two barcodes. The synthesis distance $d(b_1, b_2)$ is the Hamming distance of their synthesis schedules $s(b_1)$ and $s(b_2)$.*

In this paper, we consider the 8-neighborhood of a position to be the set of positions at which unwanted insertions are most likely to occur.

▶ **Definition 3** (8-neighborhood). *For a position (x, y) on a $w \times h$-array, the set*

$$N(x, y) := (\{(\tilde{x}, \tilde{y}) \colon |x - \tilde{x}| \leq 1 \wedge |y - \tilde{y}| \leq 1\} \setminus \{(x, y)\}) \cap \{1, \ldots, w\} \times \{1, \ldots, h\}$$

contains the 8-neighborhood around (x, y).

Now, we define a local cost function for each barcode position.

▶ **Definition 4** (local cost of a neighborhood). *Let L be a barcode layout on a $w \times h$-array. We define the* local cost *at position (x, y) as*

$$cost(L, x, y) = \sum_{(\tilde{x}, \tilde{y}) \in N(x,y)} d(L(x, y), L(\tilde{x}, \tilde{y})).$$

Finally, we are able to define the BARCODE-LAYOUT minimization problem:

▶ **Definition 5** (BARCODE-LAYOUT problem).
Input: *Set B of barcodes of length ℓ. Array dimensions $w, h \in \mathbb{N}$ where $|B| \geq wh$.*
Output: *Barcode layout L that minimizes*

$$cost(L) = \sum_{x=1}^{w} \sum_{y=1}^{h} cost(L, x, y).$$

3 Computational Complexity

Before presenting our algorithmic approaches, we would like to convey our theoretical findings on the complexity of the BARCODE-LAYOUT problem. We show that every problem in MaxSNP is L-reducible to the BARCODE-LAYOUT problem and thus, that the BARCODE-LAYOUT problem is MaxSNP-hard [29].

▶ **Definition 6** (L-reduction [29, 37]). *Consider two optimization problems A and B. A is said to be L-reducible to B if there exist two positive constants α, β and two polynomial-time functions f, g such that*

1. *for each input x to A, $f(x)$ is a valid input to B*
2. *$opt_B(f(x)) \leq \alpha \cdot opt_A(x)$*
3. *if y is a solution to B on the input $f(x)$, then $g(x, y)$ must be a valid solution to A*
4. *$|opt_A(x) - cost_A(x, g(x, y))| \leq \beta \cdot |opt_B(f(x)) - cost_B(f(x), y)|$.*

For our proof, we are going to perform an L-reduction from the path-TSP on Hamming spaces to our BARCODE-LAYOUT problem.

▶ **Definition 7** (path-TSP on Hamming spaces). *Given n cities $i = 1, 2, \ldots, n$ with coordinates $c_i \in \{0, 1\}^k$ and pairwise distances $d_{ij} = d_{Hamming}(c_i, c_j)$, find a Hamiltonian path of minimum total length.*

Trevisan [37] gave a proof for the MaxSNP-hardness of the related TSP on Hamming spaces, in which one looks for a Hamiltonian cycle instead of a Hamiltonian path. He described an L-reduction from (1,2)-TSP to TSP on Hamming spaces. It is straight-forward to apply the same technique to L-reduce (1,2)-path-TSP to path-TSP on Hamming spaces. As (1,2)-path-TSP has been shown to be MaxSNP-hard by Papadimitriou and Yannakakis [30] the following lemma holds.

▶ **Lemma 8.** *path-TSP on Hamming spaces is MaxSNP-hard.*

Having established this, we can now prove that BARCODE-LAYOUT is MaxSNP-hard by means of an L-reduction of path-TSP on Hamming spaces.

▶ **Theorem 9.** *BARCODE-LAYOUT is MaxSNP-hard.*

Proof. Consider an arbitrary instance x to path-TSP on Hamming spaces. We construct an input $f(x)$ to the BARCODE-LAYOUT problem in linear time such that every layout y to $f(x)$ is associated to some Hamiltonian path $g(x, y)$. For this purpose we transform the coordinates $c = c_1, \cdots, c_k$ of each city in x to a barcode of length $2k$ by applying the following substitutions:

$1 \mapsto AC$ with $s(AC) = 1100$

$0 \mapsto AG$ with $s(AG) = 1010$

Consequently, the synthesis distance of two barcodes is twice the Hamming distance of the original coordinates. Furthermore, the objective value $cost(y)$ of each layout is exactly twice the objective value of the corresponding Hamiltonian path $g(x, y)$, as each synthesis distance between neighboring barcodes is included twice in the cost function (see Definition 4). Thus, the transformation is isometric except for a constant factor of four. We now arrange these constructed barcodes on an $n \times 1$-array. Since the height coordinate is collapsed into a single value, the solution of the input to BARCODE-LAYOUT will provide a permutation of the input barcodes. The cost of this permutation is determined by the sum of the synthesis distances between successive barcodes. If we convert the barcodes in this permutation back to the original Hamming coordinates, we obtain an equivalent solution to path-TSP, where the sum of Hamming distances between successive Hamming coordinates is minimized. We still need to prove that this transformation is indeed an L-reduction. The application of the above substitutions represents the function f, while the application of the obtained barcode order to the original Hamming coordinates is the function g. Furthermore, the conditions 2 and 4 in Definition 6 hold true for $\alpha = \beta = 4$. So we have shown that BARCODE-LAYOUT is MaxSNP-hard, even if one of the dimensions is collapsed to a single value. ◀

4 Lower bounds

As the optimization problem is hard, we are interested in lower bounds for the best possible cost of a layout. We consider four different lower bounds and sketch the underlying ideas. In Section 6, a comparison of these bounds in terms of strength and computational effort will be presented. For ease of exposition, we assume in this section that the number of barcodes equals the number of array positions, i.e. $|B| = wh$.

LP lower bound. As mentioned before, the BARCODE-LAYOUT problem is a special case of the QAP and therefore can also be formulated as an ILP. Here, we use Lawler's linearization [22] which requires $O(wh|B|^2)$ variables. While solving these ILPs is only feasible for very small instances with state of the art solvers like Gurobi, we hope that at least the LP relaxation can be computed for larger instances.

A quadratic formulation of the BARCODE-LAYOUT minimization problem uses assignment variables x_{ijb} where (i, j) is a position on the array and $b \in B$ a barcode. The variable x_{ijb} shall be 1 if and only if barcode b is assigned to position (i, j). In a standard formulation of the QAP, quadratic terms such as $x_{ijb}x_{i'j'b'}$ are used to determine whether the barcodes b and b' are adjacent in the positions (i, j) and $(i'j')$. To linearize these quadratic terms we introduce 0/1-variables $y_{(ijb)(i'j'b')}$ such that $y_{(ijb)(i'j'b')} = x_{ijb}x_{i'j'b'}$.

Based on these variables, we obtain the following formulation:

$$\min \quad \sum_{i=1}^{w}\sum_{j=1}^{h}\sum_{b\in B}\sum_{\substack{b'\in B,\\ b'\neq b}}\sum_{\substack{(i',j')\\ \in N(i,j)}} d(b,b')\cdot y_{(ijb)(i'j'b')} \tag{1}$$

$$\text{s. t.} \quad \sum_{b\in B} x_{ijb} = 1 \qquad \forall i\in\{1,\dots,w\},\forall j\in\{1,\dots,h\} \tag{2}$$

$$\sum_{i=1}^{w}\sum_{j=1}^{h} x_{ijb} = 1 \qquad \forall b \in B \tag{3}$$

$$\sum_{i=1}^{w}\sum_{j=1}^{h}\sum_{b\in B}\sum_{\substack{b'\in B,\\ b'\neq b}}\sum_{\substack{(i',j')\\ \in N(i,j)}} y_{(ijb)(i'j'b')} = 2m \tag{4}$$

$$x_{ijb} + x_{i'j'b'} - y_{(ijb)(i'j'b')} \leq 1 \qquad \begin{matrix}\forall i\in\{1,\dots,w\},\forall j\in\{1,\dots,h\},\\ \forall b\neq b'\in B,\forall (i',j')\in N(i,j)\end{matrix} \tag{5}$$

$$x_{ijb} + x_{i'j'b'} - 2\cdot y_{(ijb)(i'j'b')} \geq 0 \qquad \forall b\neq b'\in B,\forall (i',j')\in N(i,j) \tag{6}$$

$$x_{ijb} \in \{0,1\}, y_{(ijb)(i'j'b')} \in \{0,1\} \qquad \begin{matrix}\forall i\in\{1,\dots,w\},\forall j\in\{1,\dots,h\},\\ \forall b\neq b'\in B,\forall (i',j')\in N(i,j)\end{matrix} \tag{7}$$

Equation 2 ensures that every position (i, j) of the array is assigned with exactly one barcode and Equation 3 ensures that each barcode is assigned to exactly one position. Equation 4 says that exactly $2m$ of the $y_{(ijb)(i'j'b')}$ variables are set to 1, where m is defined as the number of neighborhood relations (vertical, horizontal and diagonal) in an array of size $w \times h$:

$$m := 2\cdot(w-1)\cdot(h-1) + w\cdot(h-1) + (w-1)\cdot h \tag{8}$$

Finally, Equations 5 and 6 couple the x and y variables. Equation 5 models the implication $(x_{ijb} = 1 \wedge x_{i'j'b'} = 1) \Rightarrow y_{(ijb)(i'j'b')} = 1$ and Equation 6 models the implication $y_{(ijb)(i'j'b')} = 1 \Rightarrow (x_{ijb} = 1 \wedge x_{i'j'b'} = 1)$. Consequently, $y_{(ijb)(i'j'b')} = x_{ijb}x_{i'j'b'}$ as desired.

Gilmore-Lawler bound. In the field of QAPs, a classical lower bound is the so-called Gilmore-Lawler bound (GLB) [9, 22]. We adapt it to the special case of the BARCODE-LAYOUT problem. In a first step, we compute for each barcode the neighborhood cost for the best neighbors it can possibly have. The values differ depending on the position of the barcode on the array: in a corner, the barcode chooses 3 neighbors; in a border position 5 neighbors and in the middle 8 neighbors. That gives three coefficients for each barcode b: $l_b^{(3)}$ as the optimal cost with 3 neighbors, $l_b^{(5)}$ with 5 and $l_b^{(8)}$ with 8 neighbors. We then have to decide for each barcode whether it is best placed in a corner, at the border or in the middle.

We formulate this problem as an ILP. The binary variables $x_b^{(3)}, x_b^{(5)}$ and $x_b^{(8)}$ indicate for barcode b whether it is best placed in a corner, at the border or in the middle. We then solve a linear assignment problem to choose the optimal position for each barcode with respect to the coefficients $l_b^{(3)}, l_b^{(5)}$ and $l_b^{(8)}$. The following ILP shows how we compute the Gilmore-Lawler bound:

$$\min \quad \sum_{b \in B} \left(l_b^{(3)} \cdot x_b^{(3)} + l_b^{(5)} \cdot x_b^{(5)} + l_b^{(8)} \cdot x_b^{(8)} \right) \tag{9}$$

$$\text{s. t.} \quad \sum_{b \in B} x_b^{(3)} = 4 \tag{10}$$

$$\sum_{b \in B} x_b^{(5)} = 2 \cdot (w-2) + 2 \cdot (h-2) \tag{11}$$

$$\sum_{b \in B} x_b^{(8)} = (w-2) \cdot (h-2) \tag{12}$$

$$x_b^{(3)} + x_b^{(5)} + x_b^{(8)} = 1 \qquad \forall b \in B \tag{13}$$

$$x_b^{(3)}, x_b^{(5)}, x_b^{(8)} \in \{0,1\} \qquad \forall b \in B \tag{14}$$

We take into account how many barcodes with 3 (Equation 10), 5 (Equation 11) or 8 (Equation 12) neighbors we need for a valid layout. We also ensure that each barcode is assigned to exactly one position (Equation 13).

Kahng bound. A simple combinatorial idea is to select for each barcode the 8 closest barcodes in the set with respect to the synthesis distance. Since the border and corner positions have fewer neighbors, we have to discard a certain number of values. With m defined as in Equation 8, we then keep only the $2m$ values with the smallest weights. It is clear that the total weight cannot exceed the cost of an optimal layout and therefore is a lower bound. In the context of microarray design, this bound has been described by Kahng et al. [15] for a 4-neighborhood. It will be referred to as the *Kahng* bound in the following.

b-matching bound. Inspired by the 2-matching relaxation of the TSP, we developed a fourth bound. Let H be an undirected complete graph on the barcode set with the synthesis distance as edge weights. A subset of the edges is chosen such that each node has a degree of at least 3 and at most 8. The number of edges chosen is equal to the number of neighborhood relations on the array, i.e. m as in Equation 8. Instead of letting each barcode freely choose its eight closest neighbors as edges, we now ensure that neighboring positions must choose their edges consistently. The resulting bound is twice the sum of the edge weights chosen.

The following ILP shows how we compute the b-matching bound.

$$\min \quad 2 \cdot \sum_{b \in B} \sum_{\substack{b' \in B, \\ b' \neq b}} d(b,b') \cdot x_{bb'} \tag{15}$$

$$\text{s. t.} \quad \sum_{b \in B} \sum_{\substack{b' \in B, \\ b' \neq b}} x_{bb'} = m \tag{16}$$

$$\sum_{\substack{b' \in B, \\ b' \neq b}} x_{bb'} \leq 8 \qquad \forall b \in B \tag{17}$$

$$\sum_{\substack{b' \in B, \\ b' \neq b}} x_{bb'} \geq 3 \qquad \forall b \in B \tag{18}$$

$$x_{bb'} \in \{0,1\} \qquad \forall b \neq b' \in B \tag{19}$$

The variables $x_{bb'}$ indicate neighborhood relations: $x_{bb'} = 1$ if and only if barcodes b and b' are connected by an edge. The number of chosen edges is restricted by Equation 16 where m is the number of edges defined in Equation 8. Equations 17 and 18 ensure the node degrees.

5 Heuristic Approaches

Since we have shown that the BARCODE-LAYOUT problem is MaxSNP-hard, we decided to focus on developing heuristic approaches.

LEXSORT. We first consider a sorting based algorithm, which sorts the input lexicographically and subsequently fills the layout with barcodes column by column. This ensures that vertically neighboring barcodes have a maximum common prefix. These barcodes will be synthesized identically for the length of their common prefix. The runtime of this heuristic is dominated by sorting the input. This approach is therefore applicable to large input sets. However, it has several shortcomings in terms of layout quality. First, it only considers the common prefix of barcodes, which can lead to many missed opportunities for good neighbors, which are synthesised differently in the first few cycles. Second, it only optimizes vertically, leaving the possibility of highly sub-optimal neighbors in all other directions.

Greedy Algorithms. Kahng et al. [15] describe a greedy algorithm (which they call ROW-EPITAXIAL) and show that it gives promising results. We adapted their algorithm to our problem by making minor changes. The algorithm fills the layout column by column (originally row by row). For each position (x, y), the algorithm selects some yet unassigned barcode b that minimizes the *local synthesis distance*

$$d(L, x, y, b) = \sum_{(\tilde{x},\tilde{y})\in N(x,y)} d(b, L(\tilde{x}, \tilde{y}))$$

which describes the (partial) cost of placing the barcode b at position (x, y). When evaluating the sum, we only consider neighboring positions to which we already assigned some barcode in a previous step. When considering the i-th position (x, y), the algorithm needs to calculate the local synthesis distance for *each* of the $|B| - i + 1$ yet unassigned barcodes b. As a result, the algorithm must perform $\mathcal{O}(wh|B|)$ calculations of the local synthesis distance, which can be unpleasantly large in practice. For this reason, the original ROW-EPITAXIAL algorithm suggested by Kahng et al. had to use a limited lookahead in order to produce results within a reasonable time. Their idea was to select the best unassigned barcode b from the first $\leq 20,000$ unassigned barcodes in lexicographic order. This reduces the number of local synthesis distance evaluations to $\mathcal{O}(wh)$, since 20,000 is a constant.

In contrast to Kahng et al., we used an unlimited lookahead (i.e. we chose b from *all* unassigned barcodes). This was achieved by using GPU parallelization.

2-OPT Local Search. We developed another algorithm to improve existing layouts based on the well-known local search principle. Starting with some initial layout, we iteratively determine pairs of barcodes that can be swapped to improve the layout cost. In each iteration, we first determine the current best swap partner for each barcode on the array. Improving swaps are then performed sequentially, starting with the one that gives the best improvement. Elements that have already been repositioned in this iteration, or are in the neighborhood of such an element, are not swapped. When there are no more profitable swaps, a local minimum is reached and the algorithm terminates. It is straight-forward to extend this idea to the more general situation where we have more barcodes than positions on the layout.

In the first iteration of the local search, we have to calculate the effect of swapping *each* barcode on the layout with *every other* barcode in B. Thus, the computational effort in this iteration grows like $\mathcal{O}(wh|B|)$. In subsequent iterations we can take advantage of the fact that new improving swaps can only occur for swapped barcodes or in their neighborhood.

Table 1 Types of barcode sets used in our experiments. All barcodes have length 34 and have been sampled uniformly at random by rejection sampling as sketched in Section 1. Note that $4\ell = 136$ is an upper bound on synthesis cycles required but in practice much shorter schedules can be achieved.

Set name	Constraints
`random`	no additional constraints
`GC`	GC content is 40-60 %
`maxCycles`	number of synthesis cycles is less than 93
`no repeats`	no homopolymers and no repeats of length ≥ 3
`constrained`	GC + maxCycles + no repeats
`distance`	constrained + pairwise SL distance at least 9

Genetic Algorithm. In a further effort to improve the quality of layouts, we have developed a genetic algorithm. This approach models biological evolution by modifying a population of 1024 layouts using a combination of selection, crossover and mutation, hopefully converging towards a local optimum [13, 14]. We use tournament selection [12] with 8 participants to select a subset of the population for reproduction. The selected couples then produce two offspring using a crossover operator that is based on the partially mapped crossover operator [39], but favors barcode placements which already have a low local cost function within their neighborhood. The idea of using local cost information for an optimized crossover operator comes from [28] and [10]. Finally, a simple swap operator is applied to these offspring to improve genetic diversity.

6 Experimental Results

In this section, we evaluate the solution quality of our algorithms. For this purpose, we experiment with multiple barcode sets that differ by several additional constraints (see Table 1).

6.1 Comparison of Lower Bounds

We start by comparing the quality of the lower bounds described in Section 4.

Experimental setup. We sampled several subsets from size 10×10 to size 768×1024 from the barcode set *distance* and evaluated the four lower bounds on each subset. To solve the (I)LPs, we used Gurobi in version 9.52.

Results and discussion. Table 2 shows the results of the experiment. We see that the bounds calculated with the (I)LP solver can only be computed for small instances within reasonable time. The LP bound turns out to be much weaker than the other bounds. In contrast, the Kahng and GLB bounds are much stronger and can be evaluated for large instances within reasonable time. The b-matching is only marginally better than the Kahng bound but scales less well. Thus, we decided to use the Kahng bound in the following experiments.

6.2 Expected Cost of Random Layouts

To get another baseline for the heuristics, we calculated the expected cost of a random layout, where a random barcode is selected from B and assigned to each array position.

Table 2 Lower bounds for different array sizes and barcode sets of size $w \cdot h$ of type distance. Entries with NA stand for instances which we could not solve within reasonable time and memory.

method	10×10	15×15	20×20	100×100	768×1024
LP	13,216	30,120	NA	NA	NA
GLB	**21,344**	**48,202**	**84,988**	1,881,900	119,211,464
Kahng	20,964	47,744	84,376	1,884,112	**119,215,966**
b-matching	21,032	47,852	84,676	**1,884,904**	NA

Experimental setup. To calculate the expected layout cost $E(cost)$, we first need to calculate the expected synthesis distance $E(d)$ between two barcodes sampled uniformly at random from B. This value depends on the specific barcode set B and can be calculated by summing over all barcode pairs of this set. We then calculate the expected layout cost using the formula $E(cost) = 2m \cdot E(d)$ with m as defined in Equation 8.

Results and discussion. For the given set of type distance with 768×1024 barcodes, this resulted in $E(cost) \approx 254,498,050$ for a 768×1024 array. This baseline was also empirically confirmed as the mean cost of 10 randomly generated layouts was $254,485,241.6$ with a standard deviation of $14,378.1$. We observe that the expected cost is about twice as large as the Kahng bound.

6.3 Comparison of Heuristics

Next, we investigated the quality of our heuristics.

Experimental setup. We ran each heuristic ten times on the barcode set of type distance with 768×1024 barcodes and the same layout dimension (thus without any excess on barcodes) and determined the average layout cost. Since the genetic algorithm and the 2-OPT local search require initial layouts, we tested these algorithms first with random initial layouts and afterwards with pre-optimized layouts produced by the unlimited Greedy Algorithm. For each heuristic, we calculated the percentage improvement over the expected layout cost (gain) and the gap to Kahng's lower bound (gap).

Table 3 Gain (%) gives the improvement over the expected cost of a random layout in percent. Gap (%) shows the optimality gap to the Kahng lower bound in percent.

Algorithm	Average Cost	gain (%)	gap (%)
Lower bound (Kahng)	$119,215,966$	—	—
Random layout (expected value)	$254,498,050$	—	113.48
LEXSORT	$218,300,868$	14.22	83.11
Greedy (lookahead 20000)	$178,342,888$	29.90	49.60
Greedy (unlimited)	$159,937,955$	37.16	34.16
2-OPT Local Search (random initial layout)	$181,718,241.2$	28.60	52.43
2-OPT Local Search (pre-optimized initial layout)	$159,887,640.4$	**37.18**	**34.12**
Genetic Algorithm (random initial layouts)	$203,261,559$	20.13	70.50
Genetic Algorithm (pre-optimized initial layout)	$163,691,770$	35.70	37.31

Table 4 Average runtimes for sequential CPU or parallelized GPU implementations of selected algorithms in Table 3. Sequential runtimes for 2-OPT were estimated from the sequential runtime of a single iteration.

Algorithm	CPU (sequential)	GPU (parallelized)
LEXSORT	0.39 sec	-
Greedy (unlimited)	7.86 hours	94.1 sec
2-OPT Local Search (random initial layout)	46.4 days	15.6 min
2-OPT Local Search (pre-optimized initial layout)	3.7 days	89.5 sec

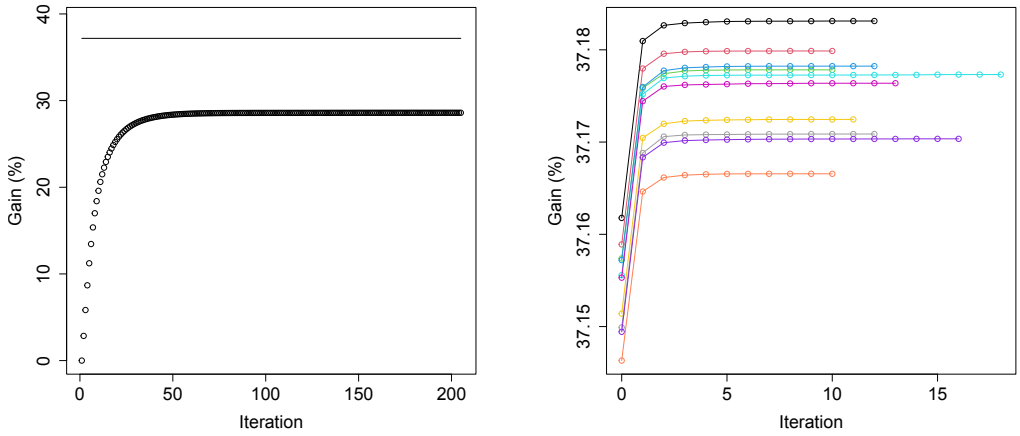
Results and discussion. Table 3 and Table 4 show the results. We highlight some essential observations.

- We observe that the unlimited Greedy Algorithm produces solutions of high quality, which could be improved marginally by a successive 2-OPT local search at the cost of small additional runtime.
- Interestingly, starting the 2-OPT local search with a random initial layout produces layouts that are local optima, but far worse than the layout produced by our greedy algorithm. This suggests that the local optima of the 2-OPT neighborhood relation differ drastically in their quality.
- Looking into the 2-OPT local search in more detail, we observe that it improved a random initial layout by about 28.60% in 204.1 iterations on average (see Figure 3a) and thus performs better than LEXSORT, but much worse than the unlimited Greedy Algorithm. Using a pre-optimized layout generated by the unlimited Greedy Algorithm, a local optimum was quickly reached after only 12.4 iterations on average, as can be seen in Figure 3b.
- Unfortunately, the Genetic Algorithm could not further reduce the cost of layouts produced by the unlimited Greedy Algorithm. Instead, we observed increasing costs within the first generations and a plateau afterwards. This behaviour is probably caused by the mutation operator, which is applied after each generation. This operator swaps randomly selected barcodes in order to maintain sufficient genetic diversity throughout the population. In highly optimized layouts, this tends to destroy good neighborhoods and create more average ones, increasing the overall cost of the layout.
- GPU parallelization greatly accelerates our algorithms compared to a sequential CPU implementation. We estimate the speedup factor of the 2-OPT algorithm to be more than 4000. The Greedy Algorithm was accelerated by a factor of about 300.

6.4 Barcode Sets with Excess

Our experiments suggest that the greedy algorithm provides a significant improvement in layout cost when compared to random layouts. However, a typical weakness for greedy algorithms is that the choices made later in the process get worse as there are fewer options left. To address this issue, we run an additional experiment where we provide more barcodes to choose from than the Greedy Algorithm needs for the layout.

Experimental setup. Starting with 768×1024 barcodes of type `random`, we iteratively increased the barcode excess by including more barcodes. In doing so, we generated barcode sets that are 1×, 1.5×, 2×, 2.5×, 3×, 3.5×, and 4× as large as the initial set. We ran the unlimited Greedy algorithm 10 times on each set and determined the average layout cost.



(a) The gain of the layout cost compared to a random layout in each iteration of the 2-OPT algorithm in ten runs (points) and the best known solution (line).

(b) The gain of the layout cost compared to a random layout in each iteration of the 2-OPT algorithm, starting with ten different pre-optimized layouts (represented by different colors).

Figure 3 The performance of local-search based 2-OPT.

Results and discussion. Figure 4 shows the average layout cost of our greedy heuristic for barcode sets with multiple excess factors. As expected, there is a clear decrease in cost as the size increases. Even a small excess factor of 1.5 has an effect: The cost decreases by about 10.71% compared to using the original set of 768×1024 barcodes (without any excess). The cost decreases by up to 43.42% compared to the baseline as the size of the barcode set increases.

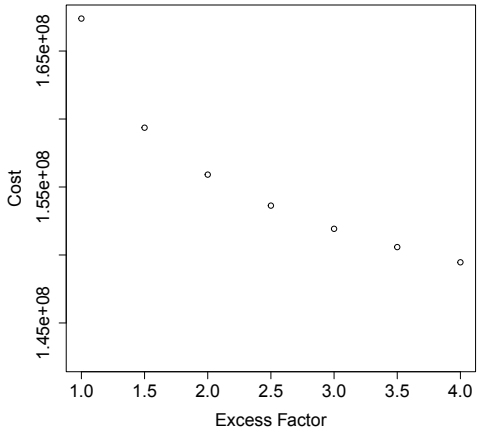


Figure 4 The average cost value for each barcode set size used.

We conclude that even a small excess of barcodes can largely improve the layout quality. Runtime increases linearly with the barcode set size: The runtime for the set of $4 \cdot w \cdot h$ barcodes is 386 seconds and thus about 4 times as long as for the set of $w \cdot h$ barcodes with 94 seconds. Our observations show that our implementation scales well and is even suitable for next-generation microarrays with four million positions.

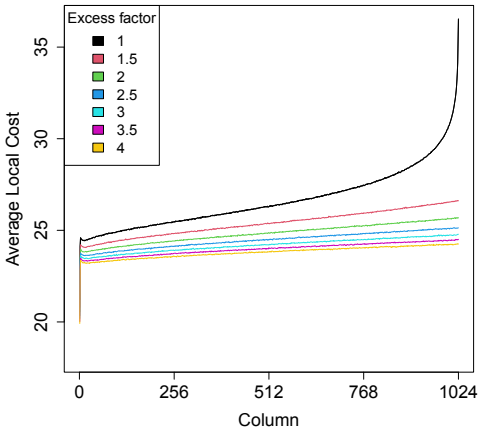


Figure 5 Average local cost of every column in the layout for each barcode set size used.

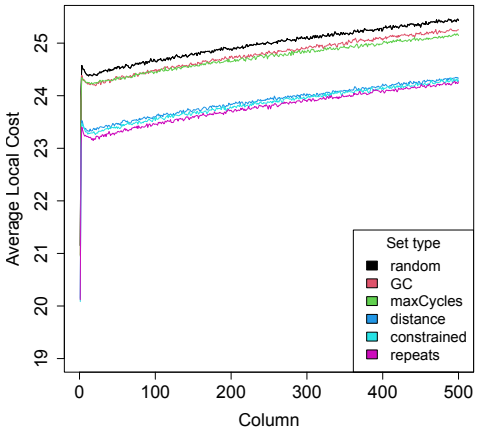


Figure 6 Average local cost of every column in the layout for each barcode set type used.

6.5 Distribution of Layout Cost

Next we analyse how much each layout position (x, y) contributes to the total layout cost. To do this, we first normalized the local cost (see Definition 4) of each position (x, y) by the number of its neighbors (which is either 3, 5, or 8). This allows us to fairly compare the contributions of each layout position regardless of their number of neighbors. For the matter of a clear presentation, we show the normalized local costs averaged over each column y. Thus, the *average local cost* of column y is defined as

$$\frac{1}{w}\sum_{x=1}^{w}\frac{cost(L,x,y)}{|N(x,y)|}.$$

The larger this value, the more the y-th column contributes to the layout cost.

Experimental setup. We repeated the previous experiment, now calculating the individual average local costs for each column.

Results and discussion. Figure 5 shows the results of this experiment. We observe a rapid increase in cost in the last columns, when the greedy algorithm has almost no freedom of choice anymore. If we use more barcodes than necessary, this effect disappears. Instead, we see an almost linear change in quality with a slope that decreases with larger barcode sets. These observations confirm our previous conclusions.

6.6 Influence of Barcode Set Type

As mentioned earlier, barcode sets are usually constrained for technical and other reasons. We therefore carried out a final experiment to assess whether our Greedy Algorithm behaves differently on constrained barcode sets.

Experimental setup. We used a barcode set of each type mentioned in Table 1 and evaluate the average layout cost of the unlimited Greedy Algorithm for an array of size 400×500. Each barcode set contained 800,000 barcodes. Similar to the previous experiment, we plot the average local cost of each individual column.

Results and discussion. Figure 6 shows the results. There are two distinct groups: The first group with the higher costs consists of the `random` set, the `GC` set and the `maxCycles` set. The `distance` set, the `constrained` set and the `no repeats` set form the second group with the lower cost values. We conclude that constraining the GC content and the number of synthesis cycles has little effect on the barcodes, which is why these sets behave almost the same as the `random` set. The exclusion of repeats greatly reduces the set of valid barcodes. This means that the average synthesis distance between barcodes is significantly reduced, which translates into lower costs for the generated layouts.

We conclude that using certain constraints on the barcode sets doesn't have a negative impact on the cost of the layout. On the contrary, the cost even improves compared to a random set of barcodes.

7 Summary

In this paper we provided a computational study on the `BARCODE-LAYOUT` problem that arises in the synthesis of DNA microarrays by photolithography. Experiments on instances of typical size demonstrate that the resulting combinatorial optimization problems are very hard to solve. We obtained the best results with a greedy approach followed by 2-OPT local search which improved on a random layout by 37%. However, the optimality gap with respect to the best lower bound is also relatively large at 34%. We suspect that the available lower bounds are quite weak so that the actual solutions are not too far away from the unknown optimum. Due to the long runtimes of the sequential versions of the Greedy and 2-OPT algorithms, it was crucial to implement variants that are accelerated on a GPU.

Our attempts to further improve the greedy solutions using stochastic meta-heuristics has been rather disappointing. Exemplarily we reported non-competitive results with a genetic algorithm. Future work could explore other meta-heuristics in more detail.

One important finding is that we can benefit greatly from selecting barcodes from a large pool of candidates. This can improve solutions by up to 43% compared to random layouts. Additionally, experiments with realistic barcode sets that meet practical side constraints result in even better layouts than those with randomly chosen barcodes.

References

1 James L Banal, Tyson R. Shepherd, Joseph Berleant, Hellen Huang, Miguel Reyes, Cheri M. Ackermann, Paul C. Blainey, and Mark Bathe. Random access DNA memory using boolean search in an archival file storage system. *Nature Materials*, 21:1272–1280, 2021. doi:10.1038/s41563-021-01021-3.

2 Hyo-eun C Bhang, David A Ruddy, Viveksagar Krishnamurthy Radhakrishna, Justina X Caushi, Rui Zhao, Matthew M Hims, Angad P Singh, Iris Kao, Daniel Rakiec, Pamela Shaw, Marissa Balak, Alina Raza, Elizabeth Ackley, Nicholas Keen, Michael R Schlabach, Michael Palmer, Rebecca J Leary, Derek Y Chiang, William R Sellers, Franziska Michor, Vesselina G Cooke, Joshua M Korn, and Frank Stegmeier. Studying clonal dynamics in response to cancer therapy using high-complexity barcoding. *Nature Medicine*, 21:440–448, 2015. doi:10.1038/nm.3841.

3 Tilo Buschmann and Leonid Bystrykh. Levenshtein error-correcting barcodes for multiplexed DNA sequencing. *BMC bioinformatics*, 14:272, September 2013. doi:10.1186/1471-2105-14-272.

4 Sérgio A. de Carvalho Jr. and Sven Rahmann. Microarray layout as quadratic assignment problem. In *German Conference on Bioinformatics*, pages 11–20. Gesellschaft für Informatik e.V., Bonn, 2006.

5 Sérgio A. de Carvalho Jr. and Sven Rahmann. Better genechip microarray layouts by combining probe placement and embedding. *Journal of bioinformatics and computational biology*, 6(3):623–641, 2008. doi:10.1142/s0219720008003576.

6 Paul Igor Costea, Joakim Lundeberg, and Pelin Akan. TagGD: Fast and accurate software for DNA tag generation and demultiplexing. *PLoS ONE*, 8(3):e57521, 2013. doi:10.1371/journal.pone.0057521.

7 Brant C. Faircloth and Travis C. Glenn. Not all sequence tags are created equal: Designing and validating sequence identification tags robust to indels. *PLOS ONE*, 7(8):e42543, 2012. doi:10.1371/journal.pone.0042543.

8 Gerd Finke, Rainer E. Burkard, and Franz Rendl. Quadratic assignment problems. In Silvano Martello, Gilbert Laporte, Michel Minoux, and Celso Ribeiro, editors, *Surveys in Combinatorial Optimization*, volume 132 of *North-Holland Mathematics Studies*, pages 61–82. North-Holland, 1987. doi:10.1016/S0304-0208(08)73232-8.

9 Paul C. Gilmore. Optimal and suboptimal algorithms for the quadratic assignment problem. *Journal of the Society for Industrial and Applied Mathematics*, 10(2):305–313, 1962. doi:10.1137/0110022.

10 John Grefenstette, Rajeev Gopal, Brian Rosmaita, and Dirk Van Gucht. Genetic algorithms for the traveling salesman problem. In *Proceedings of the 1st International Conference on Genetic Algorithms*, January 1985.

11 Sridhar Hannenhalli, Earl Hubell, Robert Lipshutz, and Pavel A. Pevzner. Combinatorial algorithms for design of DNA arrays. *Adv Biochem Eng Biotechnol.*, 77:1–19, 2002. doi:10.1007/3-540-45713-5_1.

12 Ahmad Hassanat, Khalid Almohammadi, Esra'a Alkafaween, Eman Abunawas, Awni Hammouri, and V. B. Surya Prasath. Choosing mutation and crossover ratios for genetic algorithms— a review with a new dynamic approach. *Information*, 10(12), 2019. doi:10.3390/info10120390.

13 John Holland. *Adaptation in natural and artificial systems*. The MIT Press, 1975.

14 John H. Holland. Genetic algorithms and adaptation. In Oliver G. Selfridge, Edwina L. Rissland, and Michael A. Arbib, editors, *Adaptive Control of Ill-Defined Systems*, pages 317–333. Springer US, Boston, MA, 1984. doi:10.1007/978-1-4684-8941-5_21.

15 Andrew B. Kahng, Ion I. Măndoiu, Pavel A. Pevzner, Sherief Reda, and Alexander Z. Zelikovsky. Scalable heuristics for design of DNA probe arrays. *Journal of computational biology : a journal of computational molecular cell biology*, 11(2-3):429–447, 2004.

16 Justus M. Kebschull and Anthony M. Zador. Cellular barcoding: lineage tracing, screening and beyond. *Nature Methods*, 15:871–879, 2018. doi:10.1038/s41592-018-0185-x.

17 Allon M. Klein, Linas Mazutis, Ilke Akartuna, Naren Tallapragada, Adrian Veres, Victor Li, Leonid Peshkin, David A. Weitz, and Marc W. Kirschner. Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. *Cell*, 161(5):1187–1201, 2015. doi:10.1016/j.cell.2015.04.044.

18 Tjalling C. Koopmans and Martin Beckmann. Assignment problems and the location of economic activities. *Econometrica*, 25(1):53–76, 1957.

19 Manoj Kumar, Aryabartta Sahu, and Pinaki Mitra. A comparison of different metaheuristics for the quadratic assignment problem in accelerated systems. *Applied Soft Computing*, 100:106927, 2021. doi:10.1016/j.asoc.2020.106927.

20 Vamsi Kundeti and Sanguthevar Rajasekaran. On the hardness of the border length minimization problem on a rectangular array. *International Journal of Foundations of Computer Science*, 21(6):1089–1100, 2010. doi:10.1142/S0129054110007751.

21 Vamsi Kundeti, Sanguthevar Rajasekaran, and Hieu Dinh. Border length minimization problem on a square array. *Journal of Computational Biology*, 21(6):446–455, 2014. doi:10.1089/cmb.2013.0127.

22 Eugene L. Lawler. The quadratic assignment problem. *Management Science*, 9:586–599, 1963. doi:10.1287/mnsc.9.4.586.

23 Vladimir I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics doklady*, 10(8):707–710, 1966.

24 Cindy Y. Li, Prudence W. H. Wong, Qin Xin, and Fencol C. C. Yung. Approximating border length for DNA microarray synthesis. In Manindra Agrawal, Dingzhu Du, Zhenhua Duan, and Angsheng Li, editors, *Theory and Applications of Models of Computation*, pages 410–422. Springer Berlin Heidelberg, 2008.

25 Jory Lietard, Adrien Leger, Yaniv Erlich, Norah Sadowski, Winston Timp, and Mark Somoza. Chemical and photochemical error rates in light-directed synthesis of complex DNA libraries. *Nucleic Acids Research*, 49, June 2021. doi:10.1093/nar/gkab505.

26 Yang Liu, Mingyu Yang, Yanxiang Deng, Graham Su, Archibald Enninful, Cindy C. Guo, Toma Tebaldi, Di Zhang, Dongjoo Kim, Zhiliang Bai, Eileen Norris, Alisia Pan, Jiatong Li, Yang Xiao, Stephanie Halene, and Rong Fan. High-spatial-resolution multi-omics sequencing via deterministic barcoding in tissue. *Cell*, 183(6):1665–1681.e18, 2020. doi:10.1016/j.cell.2020.10.026.

27 Eliane Maria Loiola, Nair Maria Maia de Abreu, Paulo Oswaldo Boaventura-Netto, Peter Hahn, and Tania Querido. A survey for the quadratic assignment problem. *European Journal of Operational Research*, 176(2):657–690, 2007. doi:10.1016/j.ejor.2005.09.032.

28 Chilukuri K. Mohan. Selective crossover: towards fitter offspring. In *Proceedings of the 1998 ACM symposium on Applied Computing*, SAC '98, pages 374–378, New York, NY, USA, February 1998. Association for Computing Machinery. doi:10.1145/330560.330842.

29 Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43(3):425–440, December 1991. doi:10.1016/0022-0000(91)90023-X.

30 Christos H. Papadimitriou and Mihalis Yannakakis. The traveling salesman problem with distances one and two. *Mathematics of Operations Research*, 18(1):1–11, 1993.

31 Panos M. Pardalos, Franz Rendl, and Henry Wolkowicz. The quadratic assignment problem: A survey and recent developments. In *Quadratic Assignment and Related Problems*, volume 16 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 1–42. AMS, Providence, RI, 1994.

32 Alexandru Popa, Prudence W. H. Wong, and Fencol C. C. Yung. Hardness and approximation of the asynchronous border minimization problem. In Manindra Agrawal, S. Barry Cooper, and Angsheng Li, editors, *Theory and Applications of Models of Computation*, pages 164–176. Springer Berlin Heidelberg, 2012.

33 William H Press. Fast trimer statistics facilitate accurate decoding of large random DNA barcode sets even at large sequencing error rates. *PNAS Nexus*, November 2022. doi:10.1093/pnasnexus/pgac252.

34 Sartaj Sahni and Teofilo Gonzalez. P-complete approximation problems. *J. ACM*, 23(3):555–565, 1976. doi:10.1145/321958.321975.

35 Alejandro A. Schäffer and Mihalis Yannakakis. Simple local search problems that are hard to solve. *SIAM Journal on Computing*, 20(1):56–87, 1991. doi:10.1137/0220004.

36 Allyson Silva, Leandro C. Coelho, and Maryam Darvish. Quadratic assignment problem variants: A survey and an effective parallel memetic iterated tabu search. *European Journal of Operational Research*, 292(3):1066–1084, 2021. doi:10.1016/j.ejor.2020.11.035.

37 Luca Trevisan. When Hamming meets Euclid: The approximability of geometric TSP and Steiner tree. *SIAM Journal on Computing*, 30(2):475–485, 2000. doi:10.1137/S0097539799352735.

38 Céline Trébeau, Jacques Boutet de Monvel, Fabienne Wong Jun Tai, Christine Petit, and Raphaël Etournay. DNABarcodeCompatibility: an R-package for optimizing DNA-barcode combinations in multiplex sequencing experiments. *Bioinformatics*, 35(15):2690–2691, December 2019. doi:10.1093/bioinformatics/bty1030.

39 Anantkumar Umbarkar and Pranali Sheth. Crossover operators in genetic algorithms: a review. *ICTACT Journal on Soft Computing*, 6, October 2015. doi:10.21917/ijsc.2015.0150.

40 Yuqing Wang, Xi Zhang, and Zheng Wang. Cellular barcoding: From developmental tracing to anti-tumor drug discovery. *Cancer Letters*, 567:216281, 2023. doi:10.1016/j.canlet.2023.216281.

41 Johannes Wirth, Nina Huber, Kelvin Yin, Sophie Brood, Simon Chang, Celia P. Martinez-Jiminez, and Matthias Meier. Spatial transcriptomics using multiplexed deterministic barcoding in tissue. *Nature Communications*, 14(1523), 2023. doi:10.1038/s41467-023-37111-w.