Derandomized Vector Sorting

Lenwood S. Heath
heath@cs.vt.edu

Gabriel Mateescu
mateescu@cs.vt.edu

Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061-0106

Abstract

An instance of the vector sorting problem is a sequence of k-dimensional vectors of length n. A solution to the problem is a permutation of the vectors such that in each dimension the length of the longest decreasing subsequence is $O(\sqrt{n})$. A random permutation solves the problem. Here we derandomize the obvious probabilistic algorithm and obtain a deterministic $O(kn^{3.5})$ time algorithm that solves the vector sorting problem. We also apply the algorithm to a book embedding problem.

1 Introduction

We start with a simple problem. We are given a sequence $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ of ordered pairs of integers, which we will call *vectors.* If we wish to permute (sort) these vectors so that the resulting sequence has the x's (or the y's) in nondecreasing order, we may do so in $O(n \log n)$ time. But, in general, we cannot sort the vectors so that both the x's and the y's are in nondecreasing order. However, we can sort the vectors so that the longest decreasing subsequence of x's has length at most $\lceil \sqrt{n} \rceil$ and the longest decreasing subsequence of y's also has length at most $\lceil \sqrt{n} \rceil$. First sort the vectors into a sequence so that the x's are in nondecreasing order. Now partition the vectors into $\lceil n / \lceil \sqrt{n} \rceil \rceil \leq \lceil \sqrt{n} \rceil$ contiguous subsequences, each of length at most $\lceil \sqrt{n} \rceil$. Finally sort each subsequence into nondecreasing order by the y's. The algorithm requires $O(n \log n)$ time. The resulting sequence of vectors has no decreasing subsequence of x's or of y's whose length exceeds $\lceil \sqrt{n} \rceil$. We will think of such a sequence that has a guaranteed simultaneous upper bound on the length of the longest decreasing subsequence in each of its dimensions as being "nearly" sorted. For $k = 2$, it is easy to show that $\lceil \sqrt{n} \rceil$ is the best possible simultaneous upper bound.

If we generalize the above situation to a sequence of n k-dimensional vectors, then it is convenient to think of the sequence of vectors as a $k \times n$ matrix M, where each vector yields one column of M. Sorting the vectors is then done by permuting the columns of M. As it is the relative order of elements that matters, we lose no generality in the worst case by assuming that each row of M is a permutation of the interval $[1..n] = \{1, 2, \ldots, n\}$. A permutation of $[1..n]$ is called an *n-permutation* and is an element of the symmetric group S_n. A *configuration* is a $k \times n$ matrix such that each row is an n-permutation. If $k = 3$, the obvious generalization of the algorithm above requires 3 sorting passes and yields a longest decreasing subsequence of length $\leq \left\lceil n^{2/3} \right\rceil$. For general k, we can obtain a longest decreasing subsequence of length $\leq \left\lceil n^{(k-1)/k} \right\rceil$. But these straightforward algorithms do not yield asymptotically optimal simultaneous upper bounds. In fact, as we will see, we can do almost as well for moderate k as we can for $k = 2$.

Previous research on the length of increasing and decreasing subsequences in permutations suggests appropriate simultaneous upper bounds. Let M_n be the length of a longest monotonic (increasing or decreasing) subsequence in a particular sequence of n integers, not necessarily distinct. Erdös and Szekeres [6] and Ulam [20] show that every sequence of (n^2+1) integers has $M_{n^2+1} \geq n+1$. Hammersley [8] studies the behavior of M_n for random n-permutations as n goes to infinity. In particular, if p is a random n-permutation with longest decreasing subsequence of length $L(p)$, Hammersley shows that the expected value of $L(p)$ is $\lim_{n\to\infty} \mathbf{E}\left[L(p)\right] = c\sqrt{n}$, where c is some constant. Hammersley conjectured that $c = 2$. This conjecture was confirmed by Shepp and Logan [12], who prove that $c \leq 2$, and by Kerov and Versik [11], who prove that $c \geq 2$. The results of Hammersley suggest that a good simultaneous upper bound should take the form $\lceil c_k\sqrt{n}\,\rceil$, where c_k is some real number that may depend on k. The bound may only hold asymptotically, that is, as n approaches infinity. Our problem is now two-fold. First, we need to identify good values for the c_k. Second, we wish a polynomial time algorithm that sorts any $k \times n$ configuration so that the sorted configuration has no row with longest decreasing subsequence of length exceeding $\lceil c_k\sqrt{n}\,\rceil$.

Our main result is a derandomized algorithm for the vector sorting problem that requires $O(kn^{3.5})$ time. We solve the problem using the idea of probabilistic design of deterministic algorithms [1, 14, 16]. The approach involves two steps: first design a probabilistic algorithm to solve the problem; and second obtain a deterministic algorithm using derandomization. Our derandomization is based on the method of conditional expectations, applied here in a novel way to construct a permutation that sorts the input configuration.

As discussed in Section 5, the vector sorting problem has applications in circuit layout and book embedding. The problem should be contrasted with the alternative notion of sorting that arises from comparing vectors component-wise and declaring one vector u greater than another vector v exactly when u exceeds v in every component. This comparison yields a partial order and is the dominance order of computation geometry (see Atallah and Kosaraju [2]). In the vector sorting problem, the final configuration satisfies no natural partial order. Rather the property demanded of the final configuration is a more global one.

The paper is structured as follows. In Section 2, we establish the terminology, formally state the vector sorting problem, and review the relevant background. In Section 3, we derive the value for c_k and the probabilistic algorithm for solving the corresponding vector sorting problem. In Section 4, we describe the derandomization technique and present an $O(kn^{3.5})$ deterministic algorithm for the problem. Section 5 discusses circuit layout and book embedding, two applications of the vector sorting problem.

2 Problem Formulation and Preliminaries

In this section, we formally state the vector sorting problem and develop the basic related concepts.

2.1 Problem Formulation

Let p be an n-permutation. A *decreasing subsequence* of p is a sequence $p(i_1), p(i_2), \ldots, p(i_l)$ such that $p(i_1) > p(i_2) > \cdots > p(i_l)$ and $1 \leq i_1 < i_2 < \cdots < i_l \leq n$. The number l is the *length* of the decreasing subsequence. A *longest decreasing subsequence* of p is a decreasing subsequence of p whose length is maximum; call that length $\mathrm{LDS}(p)$. The set of all longest decreasing subsequences of p of length $\mathrm{LDS}(p)$ is denoted by $\mathcal{LDS}(p)$. Similarly, one defines a *longest increasing subsequence* of p, its length $\mathrm{LIS}(p)$, and the set $\mathcal{LIS}(p)$. As an example, the permutation $p = [3\ 1\ 2\ 7\ 4\ 6\ 9\ 8\ 5]$

(a) Initial Configuration, M:

$$M = \begin{bmatrix} \boxed{3} & \boxed{2} & \boxed{1} & 6 & 5 & 4 & 7 & 8 & 9 \\ \boxed{7} & 8 & 9 & \boxed{4} & 5 & 6 & \boxed{3} & \boxed{2} & \boxed{1} \end{bmatrix}$$

(b) Permuted Configuration, $M\sigma$:

$$M\sigma = \begin{bmatrix} \boxed{3} & \boxed{2} & \boxed{1} & 6 & 5 & 4 & 9 & 8 & 7 \\ \boxed{7} & 8 & 9 & \boxed{4} & 5 & 6 & \boxed{1} & 2 & 3 \end{bmatrix}$$

Figure 1: A 2×9 configuration and its permuted configuration

is a 9-permutation with $\mathrm{LDS}(p) = 3$, $\mathcal{LDS}(p) = \{[7\ 6\ 5], [9\ 8\ 5]\}$, $\mathrm{LIS}(p) = 5$, and $\mathcal{LIS}(p) = \{[1\ 2\ 4\ 6\ 8], [1\ 2\ 4\ 6\ 9]\}$.

Suppose that M is a $k \times n$ configuration. Let $p_r(M)$ be the permutation in the rth row of M, that is, $p_r(M) = [M_{r,1}\ M_{r,2}\ \dots\ M_{r,n}]$. Define $\mathrm{LDS}(M) = \max_{1 \leq r \leq k} \mathrm{LDS}(p_r(M))$, the length of the longest decreasing subsequence in any row of M. For example, Figure 1(a) shows a 2×9 configuration. One longest decreasing subsequence for each row is indicated in the figure by boxed integers. The rows have respectively $\mathrm{LDS}(p_1(M)) = 3$ and $\mathrm{LDS}(p_2(M)) = 5$. Hence $\mathrm{LDS}(M) = 5$.

Let M be a $k \times n$ configuration, and let $\sigma \in S_n$. The permutation σ acts on M on the right by permuting its columns. In particular, the element $M_{r,i}$ in row r and column i of M is mapped to row r and column $\sigma(i)$ in $M\sigma$. When σ acts on M by permuting its columns, we call σ a *column permutation of* M. Figure 1(b) illustrates the effect of applying the permutation $\sigma = [1\ 2\ 3\ 4\ 5\ 6\ 9\ 8\ 7]$ to the configuration M in Figure 1(a). The resulting configuration $M\sigma$ has $\mathrm{LDS}(p_1(M\sigma)) = \mathrm{LDS}(p_2(M\sigma)) = 3$, hence $\mathrm{LDS}(M\sigma) = 3$. The first row of $M\sigma$ in Figure 1(b) has $\mathcal{LDS}(p_1(M)) = \{[3\ 2\ 1],\ [6\ 5\ 4],\ [9\ 8\ 7]\}$, and the second row has 27 longest decreasing subsequences of length 3. One LDS in each row is indicated in the figure by boxed integers. The configuration $M\sigma$ in Figure 1 is optimal with respect to LDS.

We now assume that we have fixed the values c_k that are our simultaneous upper bounds on LDS. Formally, the problem we wish to solve is the following:

Vector Sorting Problem
INSTANCE: A $k \times n$ configuration M.
SOLUTION: An n-permutation σ such that $\mathrm{LDS}(M\sigma) \leq \lceil c_k\sqrt{n}\,\rceil$.

The optimal universal upper bound on LDS for $k \geq 2$ is no less than $\sqrt{n}$, because there exist $k \times n$ configurations for which there is no column permutation σ that yields $\mathrm{LDS}(M\sigma) < \lceil \sqrt{n} \rceil$. Indeed, construct a worst case $2 \times n$ configuration M as follows. The first row of M is the identity permutation; that is, $p_1(M) = [1\ 2\ \ldots\ n]$. The second row of M is the reverse of the identity permutation; that is, $p_2(M) = [n\ n{-}1\ \ldots\ 1]$. Let σ be any permutation in S_n. Then any decreasing subsequence in the first row of $M\sigma$ corresponds to an increasing subsequence in the second row of $M\sigma$ and vice versa. By the Erdös-Szekeres-Ulam result, the first row has either an LIS or an LDS of length $\lceil \sqrt{n} \rceil$. If the first row has an LDS of length $\lceil \sqrt{n} \rceil$, then $\mathrm{LDS}(M\sigma) \geq \lceil \sqrt{n} \rceil$. If the first row has an LIS of length $\lceil \sqrt{n} \rceil$, then the second row has an LDS of length $\lceil \sqrt{n} \rceil$ and again $\mathrm{LDS}(M\sigma) \geq \lceil \sqrt{n} \rceil$. The algorithm from Section 1 actually attains the optimal $\mathrm{LDS}(M\sigma) = \lceil \sqrt{n} \rceil$.

2.2 Permutations and Permutation Graphs

Looking ahead to the algorithm in Section 4, we will need to build up a permutation σ one value at a time. This need motivates the following definitions. For $0 \leq t \leq n$, a *t-partial permutation* σ_t is a bijective map from its domain $\mathrm{dom}(\sigma_t)$ to its codomain $\mathrm{cod}(\sigma_t)$, where $\mathrm{dom}(\sigma_t) \subseteq [1..n]$, $\mathrm{cod}(\sigma_t) \subseteq [1..n]$, and $|\mathrm{dom}(\sigma_t)| = |\mathrm{cod}(\sigma_t)| = t$. If $n \geq t' > t$, then a t'-partial permutation $\sigma_{t'}$ is an *extension* of the t-partial permutation σ_t if $\mathrm{dom}(\sigma_t) \subseteq \mathrm{dom}(\sigma_{t'})$ and $\sigma_{t'}$ restricted to $\mathrm{dom}(\sigma_t)$ equals σ_t. Note that an n-partial permutation is just an n-permutation, but a t-partial permutation for $t < n$ is not in general a t-permutation.

Let π be an arbitrary n-permutation. We say that two integers $i, j \in [1..n]$ form an *inversion* under π, denoted $i \triangleright_\pi j$, if $i < j$ and $\pi^{-1}(i) > \pi^{-1}(j)$. The *inversion relation* induced by π is $\mathcal{I}(\pi) = \{(i,j) \mid i \triangleright_\pi j\}$. The undirected graph $G_\pi = ([1..n], \mathcal{I}(\pi))$ is called the *permutation graph* of π. A *proper coloring* (chromatic decomposition) of a graph $G = (V, E)$ is a mapping from the set of vertices V to a set of colors such that all vertices having the same color form an independent set. The *chromatic number* of G, written $\chi(G)$, is the minimum number of colors needed for a proper coloring of G. A proper coloring of the permutation graph G_π has the property that no two vertices with the same color belong to the inversion relation. The chromatic number of G_π equals $\mathrm{LDS}(\pi)$. Even, Pnueli, and Lempel [7] derive an efficient algorithm for *minimum chromatic decomposition*, i.e., chromatic decomposition employing $\chi(G)$ colors, of permutation graphs. The Even–Pnueli–Lempel algorithm has time complexity $O(n\chi(G))$; we employ it to determine the chromatic number of a permutation graph.

2.3 Probabilistic Design of Deterministic Algorithms

Our deterministic algorithm is developed by derandomizing the probabilistic algorithm implied by the probabilistic method. The paradigm of the probabilistic method (see Alon, Spencer, and Erdös [1] or Motwani and Raghavan [14]) consists of defining a sample space Ω on which a discrete random variable Y, which usually represents the count of undesirable objects, is defined. The existence of at least one desirable object is proved by a demonstration that the expected value of Y satisfies $\mathbf{E}[Y] < 1$.

Our algorithm employs the classical method of conditional expectations, due to Erdös and Selfridge [5], and contributed to by Spencer [19], Raghavan [16], and others. The method involves making a series of deterministic choices that maintain the bound on the expected values of the random variables derived from Y. The derived random variables are defined on the basis of the choices that are not yet made and that are thought of as being random. To determine the choice made at each step, Raghavan [16] uses *pessimistic estimators* of the expected value that are less costly to evaluate than the expected value itself. We do the same, though in quite a different context.

3 Randomized Algorithm

In this section, we derive the constant c_k required for the Vector Sorting Problem and give a probabilistic algorithm to solve the problem. In the next section, we derive a derandomized algorithm from the randomized one.

A *random n-permutation* σ is a permutation chosen from the sample space $\Omega = S_n$ uniformly at random. In particular, every permutation in Ω has probability $1/n!$ of being chosen. For a $k \times n$ configuration M, let $\langle M \rangle = \{M\sigma \mid \sigma \in \Omega\}$ be the set of all configurations obtained from M by permuting its columns; clearly, $|\langle M\rangle| = n!$ A random n-permutation σ determines the random configuration $M\sigma$ in $\langle M\rangle$.

For the remainder of the paper, m is a fixed integer between 1 and n; we think of a desirable permutation σ as being one that makes $\mathrm{LDS}(M\sigma) < m$. We define several random variables on $\langle M\rangle$ as follows.

- $X_r = X_r(M\sigma)$ is the indicator random variable for whether row r contains a decreasing subsequence of length m; that is

$$X_r = \begin{cases} 1 & \text{if row } r \text{ of } M\sigma \text{ contains a decreasing subsequence of length } m; \\ 0 & \text{otherwise.} \end{cases}$$

- $Y = Y(M\sigma) = X_1 + X_2 + \cdots + X_k$ is the random variable that is the number of rows in $M\sigma$ that have LDS greater than or equal to m.

- $W_r = W_r(M\sigma)$ is the random variable that is the number of decreasing subsequences of length m in row r of $M\sigma$. Hence $X_r = 1$ if and only if $W_r \geq 1$.

- $Z = Z(M\sigma) = W_1 + W_2 + \cdots + W_k$ is the random variable that is the total number of decreasing subsequences of length m in all rows of $M\sigma$. By linearity of expectation, we have $\mathbf{E}[Z] = \sum_{r=1}^{k} \mathbf{E}[W_r] = k\mathbf{E}[W_1]$.

Lemma 1 *The expected number of rows of $M\sigma$ containing a decreasing subsequence of length m, $\mathbf{E}[Y]$, satisfies the bound*

$$\mathbf{E}[Y] \leq k\mathbf{E}[W_1].$$

Proof. From the definitions, $X_r \leq W_r$. Hence we have

$$\sum_{r=1}^{k} X_r = Y \leq Z = \sum_{r=1}^{k} W_r.$$

The lemma follows by applying linearity of expectation and the fact that each $\mathbf{E}[W_r] = \mathbf{E}[W_1]$. □

In his Theorem 6, Hammersley [8] proves the following result.

Lemma 2 *Suppose $m = \lceil e\sqrt{n}\,\rceil + t$ where $t \geq 0$. Then the probability that a random element $\sigma \in S_n$ has $\mathrm{LDS}(\sigma) \geq m$ satisfies the bound*

$$\mathbf{Pr}\,[\mathrm{LDS}(\sigma) \geq m] \;\leq\; \frac{e^{-2t}}{2\pi\sqrt{n}}.$$

The following theorem now provides the needed c_k's.

Theorem 3 *Let M be a $k \times n$ configuration. If $k \leq \lfloor 2\pi\sqrt{n}\,\rfloor$, then $\mathbf{E}\,[Y] < 1$ and there exists a permutation σ such that $\mathrm{LDS}(M\sigma) \leq \lceil e\sqrt{n}\,\rceil$. More generally, suppose that $k \leq \alpha n^{\beta}$, where α and β are constants. If $m = \lceil e\sqrt{n}\,\rceil + \max\{1, (\log\alpha + (\beta - 1/2)\log n)/2\}$, then there exists a permutation σ such that $\mathrm{LDS}(M\sigma) < m$.*

Proof. First suppose that $k \leq \lfloor 2\pi\sqrt{n}\,\rfloor$. Let $m = \lceil e\sqrt{n}\,\rceil + 1$, and let $t = m - \lceil e\sqrt{n}\,\rceil = 1$. Since $t > 0$, we can apply Lemma 2 to obtain

$$\begin{aligned} k\mathbf{E}\,[W_1] &\leq k\frac{e^{-2t}}{2\pi\sqrt{n}} \\ &\leq \lfloor 2\pi\sqrt{n}\,\rfloor \frac{e^{-2t}}{2\pi\sqrt{n}} \\ &< 1. \end{aligned}$$

By Lemma 1, $\mathbf{E}\,[Y] < 1$. This implies that there exists $\sigma \in S_n$ such that $Y(M\sigma) < 1$. Since Y is an integer, we have $Y(M\sigma) = 0$; that is, $\mathrm{LDS}(M\sigma) < m$ or $\mathrm{LDS}(M\sigma) \leq \lceil e\sqrt{n}\,\rceil$, as desired.

Now suppose that $k \leq \alpha n^{\beta}$, where α and β are constants. Let

$$t \;=\; \max\{1, (\log\alpha + (\beta - 1/2)\log n)/2\},$$

and let $m = \lceil e\sqrt{n}\,\rceil + t$. If $(\log\alpha + (\beta - 1/2)\log n)/2 < 1 = t$, then $k/\sqrt{n} = e^{\log\alpha + (\beta-1/2)\log n} < e^{2t}$. If $(\log\alpha + (\beta - 1/2)\log n)/2 = t$, then $k/\sqrt{n} = e^{2t}$. Since $t > 0$, we can apply Lemma 2 to obtain

$$\begin{aligned} k\mathbf{E}\,[W_1] &\leq k\frac{e^{-2t}}{2\pi\sqrt{n}} \\ &\leq e^{2t}\sqrt{n}\,\frac{e^{-2t}}{2\pi\sqrt{n}} \\ &< 1. \end{aligned}$$

As before, $\mathbf{E}\,[Y] < 1$, implying the existence of $\sigma \in S_n$ such that $\mathrm{LDS}(M\sigma) < m$, as desired. $\square$

Algorithm 1 Randomized Algorithm for the Vector Sorting Problem

RANDOMIZED_REACH_LDS(M)

$\text{LDS}(M) = \max_{1 \leq r \leq k} \left\{ \text{CHROMATIC\_DECOMPOSITION}(G_{p_r(M)}) \right\}$

while $\text{LDS} > \lceil c_k \sqrt{n} \rceil$

  do select $\sigma \in \Omega$ uniformly at random

  $\text{LDS} = \max_{1 \leq r \leq k} \left\{ \text{CHROMATIC\_DECOMPOSITION}(G_{p_r(M\sigma)}) \right\}$

return σ

The theorem suggests that we select $c_k = e$, for all $k \leq \lfloor 2\pi\sqrt{n} \rfloor$. If $k = O(n^{\beta})$ for some β, then we can select $c_k = e + \epsilon$ for any $\epsilon > 0$ and we will be able to find a solution to the Vector Sorting Problem for sufficiently large n.

The probabilistic algorithm RANDOMIZED_REACH_LDS (Algorithm 1) solves the Vector Sorting Problem under the assumptions just mentioned. It generates random permutations σ until it finds one that satisfies $\text{LDS}(M\sigma) \leq \lceil c_k \sqrt{n} \rceil$. Since $\text{LDS}(p_r(M\Sigma)) = \chi(G_{p_r(M\sigma)})$, the algorithm invokes CHROMATIC_DECOMPOSITION($G_{p_r(M\sigma)}$), the implementation of the Even–Pnueli–Lempel algorithm that returns the chromatic number of a permutation graph. From the proof of Theorem 3, it is clear that there is a tradeoff between the constant c_k and the probability that a random σ has $\text{LDS}(M\sigma) < \lceil c_k \sqrt{n} \rceil$. In fact the constant c_k can be chosen a bit larger than is required to prove the existence of σ so as to make RANDOMIZED_REACH_LDS execute in expected polynomial time.

4 Deterministic Algorithm

In this section, we derive a deterministic polynomial time algorithm for the Vector Sorting Problem. In particular, we determine the requisite $\sigma \in S_n$ one value at a time. First we choose an appropriate column s_1 of M to be the first column of $M\sigma$. In particular, this choice must guarantee that the remaining choices for σ still allow us to attain $Y(M\sigma) = 0$. Similarly, we choose an appropriate column s_2 of M to be the second column of $M\sigma$. Continuing in this fashion, the algorithm iteratively constructs the permutation σ one column at a time.

More formally, the algorithm starts with the trivial 0-partial permutation $\sigma_0 : \emptyset \to \emptyset$ and iteratively constructs t-partial permutations $\sigma_t : \mathrm{dom}(\sigma_t) \to \mathrm{cod}(\sigma_t)$ for $t = 1, 2, \ldots, n$. Each σ_t extends σ_{t-1}. Specifically, $\mathrm{cod}(\sigma_t) = [1..t]$, and the extension of σ_{t-1} to σ_t entails the selection of $s_t \in [1..n] - \mathrm{dom}(\sigma_{t-1})$ so that $\sigma_t(s_t) = t$.

Fix $m = \lceil c_k\sqrt{n}\,\rceil$, where c_k is determined in accordance with Theorem 3. For $1 \le t \le n$, assume that σ_{t-1} has been chosen and that we are contemplating our choice for σ_t. Let Y_t be the number of decreasing subsequences of length m in $M\sigma$, where $\sigma \in S_n$ is an extension of σ_{t-1}. As $Y_1 = Y$ and $\mathbf{E}[Y] < 1$, the algorithm must guarantee at each step t that $\mathbf{E}[Y_{t-1}] < 1$ implies $\mathbf{E}[Y_t] < 1$, so that finally $\mathrm{LDS}(M\sigma_n) < m$. We use the method of conditional expectations to choose each successive extension.

4.1 Method of Conditional Expectations

The required column permutation σ is constructed inductively. For $0 \le t \le n$, the t-partial permutation σ_t has domain $\mathrm{dom}(\sigma_t) = \{s_1, s_2, \cdots, s_t\}$ and codomain $\mathrm{cod}(\sigma_t) = [1..t]$. Let $\mathrm{free}(t) = [1..n] - \mathrm{dom}(\sigma_t)$, the set of columns left unassigned by σ_t. If $s \in \mathrm{free}(t-1)$, define $\sigma_{t-1}[s \to t]$ to be the t-partial permutation that extends σ_{t-1} by making $(\sigma_{t-1}[s \to t])(s) = t$. Define

$$\Omega(\sigma_{t-1}) \quad = \quad \{\sigma \in S_n \mid \sigma \text{ is an extension of } \sigma_{t-1}\}.$$

Observe that $|\Omega(\sigma_{t-1})| = (n-t)!$ and that $\Omega(\sigma_t) \subset \Omega(\sigma_{t-1})$.

Let $\langle M\rangle_t = \{M\sigma \mid \sigma \in \Omega(\sigma_{t-1})\}$ be the set of configurations from $\langle M\rangle$ obtained by permuting the columns of M with permutations from $\Omega(\sigma_{t-1})$. Define the random variables Y_t and Z_t to be the random variables identical to Y and Z respectively, except that the sample space is $\langle M\rangle_t$. Note that $Y_1 = Y$ and $Z_1 = Z$. Since $Y_t \le Z_t$, we have $\mathbf{E}[Y_t] \le \mathbf{E}[Z_t]$. We use $\mathbf{E}[Z_t]$ as our pessimistic estimator of $\mathbf{E}[Y_t]$. The following lemma expresses $\mathbf{E}[Z_t]$ in terms of conditional expectations.

Lemma 4 *Let M be a $k \times n$ configuration. Let π be a random permutation from $\Omega(\sigma_{t-1})$. For every $t \in [1..n]$, we have*

$$\mathbf{E}[Z_t] \quad = \quad (n-t+1)^{-1} \sum_{s \in \mathrm{free}(t-1)} \mathbf{E}[Z_t \mid \pi(s) = t].$$

Furthermore, if $\mathbf{E}[Z_t] < 1$*, then there exists an* $s \in \mathrm{free}(t-1)$ *such that* $\mathbf{E}[Z_t \mid \pi(s) = t] < 1$.

Proof. Recall that $|\mathrm{free}(t-1)| = n - t + 1$. Since the conditional expectation of Z_t satisfies

$$\mathbf{E}[Z_t] = \sum_{s \in \mathrm{free}(t-1)} \mathbf{Pr}[\pi(s) = t]\, \mathbf{E}[Z_t \mid \pi(s) = t]$$

and $\mathbf{Pr}[\pi(s) = t] = (n - t + 1)^{-1}$, we have

$$\mathbf{E}[Z_t] = (n - t + 1)^{-1} \sum_{s \in \mathrm{free}(t-1)} \mathbf{E}[Z_t \mid \pi(s) = t].$$

From this equation, we see that if $\mathbf{E}[Z_t] < 1$, then there exists an $s \in \mathrm{free}(t-1)$ such that $\mathbf{E}[Z_t \mid \pi(s) = t] < 1$. The lemma follows. $\square$

Theorem 5 asserts that the desired column permutation σ can be constructed inductively.

Theorem 5 *Let* M *be an arbitrary* $k \times n$ *configuration. For every* $t \in [1..n]$*, there is a* t*-partial permutation* σ_t *such that* $\mathbf{E}[Z_t] < 1$.

Proof. The theorem follows by induction on t, where Lemma 4 is applied at each inductive step to select s_t. $\square$

4.2 Estimating the Conditional Expectations

It remains to show how to compute the values $\mathbf{E}\left[Z_t \mid \pi^{-1}(t) = s\right]$ efficiently. Let $D^r_{t,s}(m)$ denote the total number of decreasing subsequences of length m in row r among all configurations $M\pi$, where $\pi \in \Omega(\sigma_{t-1}[s \to t])$. Let $D_{t,s}(m)$ be the total count of decreasing subsequences of length m over all rows among all configurations $M\pi$; that is,

$$D_{t,s}(m) = \sum_{r=1}^{k} D^r_{t,s}(m).$$

We have

$$\mathbf{E}[Z_t \mid \pi(s) = t] = \frac{D_{t,s}(m)}{(n-t)!}.$$

The function $U_t(s) = \frac{D_{t,s}(m)}{(n-t)!}$ is an estimator that satisfies $\min_{s \in \mathrm{free}(t-1)}\{U_t(s)\} \leq \mathbf{E}[Z_t]$. In the remainder of this section, we show how to find the column $s = s_t \in \mathrm{free}(t-1)$ that minimizes $U_t(s)$.

We consider all columns $s \in \mathrm{free}(t-1)$ and compute $U_t(s)$. We evaluate $D_{t,s}(m)$ after σ_{t-1} is constructed and when we need to determine σ_t. Denote by M' the $k \times n$ matrix storing the partial solution configuration constructed so far; at the beginning of step t, only the first $t-1$ columns of M' are determined. As before, let $M_{r,s}$ be the element in row r and column s of configuration M.

For each row r, we maintain two data structures: a table L_t^r and a one-dimensional array p_t^r. These contain data about the partially constructed configuration M' required to compute the expected number of decreasing subsequences of length m. We need to consider shorter subsequences of length l, where $1 \le l < m$, that are present in row r of M', as such a subsequence may be extended to a decreasing subsequence of length m. Any such subsequence of length l has a last element j, which we call the *tail* of the subsequence. For $1 \le l \le m$ and $1 \le j \le n$, the table entry $L_t^r[l][j]$ is the number of decreasing subsequences of length l in row r of M' whose tail is j. The L_t^r tables satisfy the following recurrence:

$$L_t^r[l][j] \quad = \quad \begin{cases} L_{t-1}^r[l][j] & \text{if } j \neq M_{r,s_t}; \\ \displaystyle\sum_{i=m-l+1}^{n} L_{t-1}^r[l-1][i] & \text{if } j = M_{r,s_t}. \end{cases} \tag{1}$$

For $1 \le j \le n$, the table entry $p_t^r[j]$ is the count of the numbers in row r that are not assigned in the first t steps and are smaller than j. The p_t^r arrays satisfy the following recurrence:

$$p_t^r[j] \quad = \quad \begin{cases} p_{t-1}^r[j] & \text{if } j \le M_{r,s_t}; \\ p_{t-1}^r[j] - 1 & \text{if } j > M_{r,s_t}. \end{cases} \tag{2}$$

The number of decreasing subsequences of length m that can be obtained after assigning the first t positions is given by the following theorem.

Theorem 6 *Let $t \in [1..n]$, $s \in free(t-1)$, and $\sigma_t = \sigma_{t-1}[s \to t]$. The number of decreasing subsequences of length m of a $k \times n$ configuration in the sample space $\Omega(\sigma_t)$, where $0 \le t \le (n-1)$, is given by*

$$D_{t,s}(m) =$$
$$k\binom{n-t}{m}^2 (n-m-t)! + \sum_{r=1}^{k} \sum_{l=1}^{m-1} \left[(n-m+l-t)! \sum_{j=m-l+1}^{n-l+1} L_t^r[l][j] \binom{n-t}{m-l} \binom{p_t^r[j]}{m-l} \right].$$

Proof. Since the t columns already assigned never contain a decreasing subsequence of length m, there are two ways to obtain a decreasing subsequence of length m. The first way is to form subsequences of length m out of the $n-t$ unassigned positions. There are $\binom{n-t}{m}$ choices for the positions, and these positions can contain any m of the $n-t$ numbers not assigned. The remaining $n-m-t$ numbers may be arranged in $(n-m-t)!$ ways. Thus there are $\binom{n-t}{m} \times \binom{n-t}{m} \times (n-m-t)!$ such subsequences. Looking at all k rows, the contribution to $D_{t,s}(m)$ is $k\binom{n-t}{m}^2(n-m-t)!$

The second way is to append to a subsequence of length l drawn from the first t positions a subsequence of length $m-l$ drawn from the last $n-t$ positions; we can do this if the tail j of the subsequence of length l is greater than the number of elements to be appended $(m-l)$, that is, if $m-l+1 \leq j$. For each such j, a decreasing subsequence of length m is formed out of the $L_t^r[l][j]$ subsequences of length l in the fixed part of the target, and the subsequences of length $m-l$ in the unassigned part. There are $\binom{n-t}{m-l}$ choices for the positions, and these positions can contain any of the $p_t^r[j]$ numbers less than j. The remaining $n-m+l-t$ numbers may be arranged in $(n-m+l-t)!$ ways. Thus there are

$$\sum_{j=m-l+1}^{n-l+1} L_t^r[l][j] \cdot \binom{n-t}{m-l}\binom{p_t^r[j]}{m-l}(n-m+l-t)!$$

subsequences of this kind. Adding the two contributions to $D_{t,s}(m)$ yields the desired result. □

The desired algorithm now follows from the previous discussion. The pseudocode for computing the desired σ consists of three procedures: the main procedure REACH_LDS (Algorithm 2) builds the sequence of partial permutations, ESTIMATE (Algorithm 3) computes the estimator for the partial configuration, and UPDATE (Algorithm 4) updates the L_t^r tables. At step t, the variable L contains the values in the L_{t-1}^r table, while the variable Tmp^r contains the potential L_t^r table as successive values for s_t are examined.

The following theorem summarizes our main result.

Theorem 7 *Let k and n be given. If $k \leq \lfloor 2\pi\sqrt{n} \rfloor$, then let $c_k = e$. If $k \leq \alpha n^\beta$, where α and β are constants. then let $c_k = \lceil e + \max\{1, (\log\alpha + (\beta - 1/2)\log n)/(2\lceil\sqrt{n}\rceil)\}\rceil$. Let M be a $k \times n$ configuration. Then the call* REACH_LDS(M) *returns a $\sigma \in S_n$ such that* $\mathrm{LDS}(M\sigma) \leq \lceil c_k\sqrt{n}\rceil$. *The time complexity of* REACH_LDS *is $O(kn^{3.5})$.*

Algorithm 2 Driver for Computing the Column Permutation

```
REACH_LDS(M)
▷ Input: k × n configuration M.
▷ Returns: permutation σ such that LDS(Mσ) ≤ ⌈e√n⌉
Precompute (i choose j) for 0 ≤ j ≤ i ≤ n.
σ[1..n] ← 0                  ▷ Array to hold the constructed permutation.
L ← 0                        ▷ Initial L values are unimportant.
for t ← 1 to n
    do for s ← 1 to n
           do min ← k/m! · (n choose m)
              Tmp ← L
              while σ[s] > 0
                    do s ← s + 1
              newMin ← ESTIMATE(s, t)
              if newMin < min
                then do min ← newMin
                        source ← s
       σ[source] ← t
       UPDATE(L, source, t)
       Update the p_t^r arrays        ▷ Use Equation (2)
return σ
```

Proof. The correctness of the algorithm follows from the preceding discussion. It only remains to analyze the time complexity.

Precomputing all the binomial coefficients requires $O(n^2)$ time and $O(n^2)$ space. Computing each $D_{t,s}(m)$ in function ESTIMATE takes $O(k \cdot \sqrt{n} \cdot n)$ time. Updating the Tmp table for each target takes $O(k \cdot \sqrt{n} \cdot n)$ time. The time for one inner iteration of REACH_LDS is $O(k \cdot n^{1.5})$. At the outer iteration for each t, the number of considered sources is $n - t$, so the overall time

Algorithm 3 Function to Compute the Estimator

ESTIMATE(s, t)

$\triangleright$ Input: The target t and candidate source s

$\triangleright$ Returns: Pessimistic estimator $\mathbf{E}\left[Z_t \mid \pi^{-1}(t) = s_i\right]$

UPDATE(Tmp, s, t)

$m \leftarrow 3\lceil\sqrt{n}\,\rceil + 1$ $\triangleright$ Length of the subsequence

$D_{t,s}(m) \leftarrow k \cdot \binom{n-t}{m} \cdot \binom{n-t}{m} \cdot (n-m-t)!$ $\triangleright$ Case $l = 0$

for $r \leftarrow 1$ **to** k

  do $D^r_{t,s}(m) \leftarrow 0$

    for $l \leftarrow 1$ **to** $m-1$

      do $X \leftarrow \sum_{j=m-l+1}^{n-l+1} Tmp^r[l][j] \cdot \binom{n-t}{m-l}\binom{p^r_t[j]}{m-l}$

        $D^r_{t,s}(m) \leftarrow D^r_{t,s}(m) + (n-m+l-t)! \times X$

  $D_{t,s}(m) \leftarrow D_{t,s}(m) + D^r_{t,s}(m)$ $\triangleright$ sum the count for each row

return $D_{t,s}(m)/(n-t)!$

complexity is $O(k \cdot n^{1.5} \sum_{t=1}^{n-1}(n-t)) = O(kn^{3.5})$, as claimed. $\square$

5 Applications

In this section, we present an application of the vector sorting problem to a circuit layout problem and discuss the connection between our problem and book embeddings of graphs.

5.1 A Layout Problem

Consider $2n$ modules grouped in two parallel rows of length n separated by a channel, where each module has k ports and all nets have only two terminals (two-point connections). Assume that every net must be routed in one layer only, i.e., no vias are allowed. We consider a VLSI layout problem (see, for example, [3]) in which the positions of the modules in the top row are fixed and the modules in the bottom row must be arranged so as to route the net in at most $c\sqrt{n}$ layers, where

Algorithm 4 Procedure to Update the Decreasing Subsequences Table L

UPDATE($Table, src, tar$)

$\triangleright$ Uses Equation (1) to update the L table
for $r \leftarrow 1$ to k
  do $tail \leftarrow M_{r,src}$
    $Table^r[1][tail] \leftarrow 1$
    if $tar = 1$
      then for $1 \leq j \leq n, j \neq tail$
        do $Table^r[1][j] \leftarrow 0$ $\triangleright$ $Table^r[1][j] \leftarrow 0, j \neq tail$
          for $l \leftarrow 2$ **to** m $\triangleright$ Lines other than 1
            do for $j \leftarrow 1$ **to** n
              do $Table^r[l][j] \leftarrow 0$
      else for $l \leftarrow 2$ to $m - 1$
        do $Table^r[l][tail] \leftarrow \sum_{j=tail+1}^{n-l+1} Table^r[l-1][j]$

c is a constant. We represent this problem as the problem of drawing the vertices of a bipartite graph G in two rows, one row per partition, and coloring the edges using at most $c\sqrt{n}$ colors so that any two crossing edges have distinct colors.

The vector sorting algorithm relates to the placement problem as follows. For each connected component of G there is a permutation graph and a configuration representation. A *decreasing path* of a $k \times n$ configuration M is a sequence $p_{r_1}(j_1) > p_{r_2}(j_1) > \cdots > p_{r_l}(j_l)$, where $1 \leq j_1 < j_2 < \cdots < j_l \leq n$ and $r_1, r_2, \ldots r_l \in \{1, 2, \ldots k\}$. The *longest decreasing path* of M, denoted by $LDP(M)$, is a decreasing path of M of maximum length. The minimum number of routing layers for the nets is equal to the maximum chromatic number over these permutation graphs and is also equal to the maximum LDP over the configuration. Because $LDP(M_i) \leq k \cdot \mathrm{LDS}(M_i)$, a column permutation σ_i such that $\mathrm{LDS}(M_i\sigma_i) \leq 3\sqrt{n_i}$ guarantees an upper bound on $LDP(M_i)$.

5.2 Book Embedding

Let $G = (V, E)$ be an undirected graph. A *book embedding* of G into a book of p pages has two aspects. The first aspect is a total order $<$ on V. With respect to $<$, two edges $(u, v), (x, y) \in E$, where $u < v$ and $x < y$, *cross* if either $u < x < v < y$ or $x < u < y < v$. The second aspect of a book embedding is an assignment of each edge of G to one of p pages labeled $1, 2, \ldots, p$ in such a way that no two edges assigned to the same page cross. The *pagenumber* of G is the smallest p such that G has a book embedding into a book of p pages.

Book embeddings have been studied algorithmically by numerous researchers, including Chung, Leighton and Rosenberg [4], Heath [9], Heath and Istrail [10], and Malitz [13]. In particular, Malitz [13] uses a probabilistic algorithm to prove that every $G = (V, E)$ has pagenumber $O(\sqrt{|E|})$.

Recently, Shahrokhi and Shi [18] and Shahrokhi [17] have shown how to derandomize a probabilistic algorithm that embeds a bipartite algorithms $G = (V_1, V_2, E)$ in a book of $O(\sqrt{|E|})$ pages in a separated fashion, that is, with a total order $<$ such that all the vertices in V_1 are less than all the vertices of V_2. Taking $|V_1 \cup V_2| = n$, Shahrokhi [17] presents a deterministic algorithm of time complexity $O(n^2|E|^{1.5})$ to accomplish this book embedding. He then generalizes the algorithm to a t-partite graph G to obtain a deterministic algorithm of time complexity $O(n^2|E|^{1.5})$ to embed G in $O(\sqrt{t|E|})$ pages.

With our framework, we can obtain book embeddings in the same number of pages for graphs of bounded degree. Start with a bipartite graph $G = (V_1, V_2, E)$. Let $n = |V_2|$, and let k be the maximum degree of any vertex in V_2. Choose an arbitrary fixed labeling of V_1 by labels from $[1..|V_1|]$. For each vertex in V_2, construct a k-vector of the vertices in V_1 that it is adjacent to; if the vertex has degree less than k, then the vector may be padded with repeats. From these n vectors, we can construct a $k \times n$ configuration M as usual. Applying our algorithm to M, we obtain a σ that solves the Vector Sorting Problem. This σ provides the ordering for V_2 in the book embedding of G. Row r of the resulting $M\sigma$ represents n of the edges of G. If the LDS of row r is $\mathrm{LDS}(r)$, then the edges in row r can be embedded in $\mathrm{LDS}(r)$ pages. The total number of pages required is

$$\sum_{r=1}^{k} \mathrm{LDS}(r) = O(k\sqrt{n}).$$

If k is a constant, then we obtain an $O(\sqrt{|E|})$ page book embedding in time $O(n^{3.5})$, matching Shahrokhi's result. The generalization to t-partite graphs works similarly.

Neither our result nor Shahrokhi's result immediately yield a derandomization of Malitz's general $O(\sqrt{|E|})$ bound on the pagenumber of an arbitrary graph. Both results are based on *separated book embeddings,* a concept first studied formally by Pemmaraju [15]. His development suggests that a separated book embedding for a t-partite graph can increase the pagenumber by a factor that is a function of t over the pagenumber of an optimal non-separated book embedding. As Malitz's result depends on a non-separated book embedding, we believe that attaining Malitz's general $O(\sqrt{|E|})$ bound requires derandomizing his probabilistic algorithm in a way that yields non-separated book embeddings. We conjecture that such a derandomization can be found from a closer examination of Malitz's result and the use of pessimistic estimators.

References

[1] N. Alon, J. H. Spencer, and P. Erdős, *The Probabilistic Method*, John Wiley & Sons, Inc., New York, 1992.

[2] M. J. Atallah and S. R. Kosaraju, *An efficient algorithm for maxdominance, with applications*, Algorithmica, 4 (1989), pp. 221–236.

[3] B. Berger, M. Brady, D. Brown, and F. T. Leighton, *Nearly optimal algorithms and bounds for multilayer channel routing*, Journal of the ACM, 42 (1995), pp. 500–542.

[4] F. R. K. Chung, F. T. Leighton, and A. Rosenberg, *Embedding graphs in books: A layout problem with applications to VLSI design*, SIAM J. Alg. Disc. Meth., 8 (1987), pp. 33–66.

[5] P. Erdős and J. L. Selfridge, *On a combinatorial game*, Journal of Combinatorial Theory, Series A, 14 (1973), pp. 298–301.

[6] P. Erdös and G. Szekeres, *A combinatorial problem in geometry*, Compositio Math., 2 (1935), pp. 463–470.

[7] S. Even, A. Pnueli, and A. Lempel, *Permutation graphs and transitive graphs*, Journal of the Association for Computing Machinery, 19 (1972), pp. 400–410.

[8] J. Hammersley, *A few seedlings of research*, in Proccedings of the Sixth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, Theory of Statistics, Berkeley, CA, 1972, University of California Press, pp. 345–394.

[9] L. S. Heath, *Embedding outer planar graphs in small books*, SIAM Journal on Algebraic and Discrete Methods, 8 (1987), pp. 198–218.

[10] L. S. Heath and S. Istrail, *The pagenumber of genus g graphs is O(g)*, Journal of the ACM, 35 (1992), pp. 479–501.

[11] S. V. Kerov and A. M. Versik, *Asymptotics of the Plancherel measure on the symmetric group and the limiting form of the Young tableaux*, Soviet Math. Dokl., 18 (1977), pp. 527–531.

[12] B. F. Logan and L. A. Shepp, *A variational problem for random Young tableaux*, Advances in Mathematics, 26 (1977), pp. 206–222.

[13] S. M. Malitz, *Graphs with E edges have pagenumber $O(\sqrt{E})$*, Journal of Algorithms, 17 (1994), pp. 71–84.

[14] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge University Press, New York, NY, 1995.

[15] S. V. Pemmaraju, *Exploring the Powers of Stacks and Queues via Graph Layouts*, PhD thesis, Virginia Polytechnic Institute and State University, 1992.

[16] P. Raghavan, *Probabilistic construction of deterministic algorithms: Approximating packing integer programs*, Journal of Computer and System Sciences, 37 (1988), pp. 130–143.

[17] F. Shahrokhi, *Polynomial time deterministic algorithms for page number, crossing and disjoint sets*. Submitted for publication, 1998.

[18] F. Shahrokhi and W. Shi, *Efficient deterministic algorithms for embedding graphs on books*, in Computing and Combinatorics (COCOOH '96), LNCS 1090, Springer-Verlag, 1996, pp. 162–168.

[19] J. H. Spencer, *Randomization, derandomization, and antirandomization: three games*, Theoretical Computer Science, 131 (1994).

[20] S. M. Ulam, *Monte Carlo calculations in problems of mathematical physics*, in Modern Mathematics for the Engineer: Second Series, E. F. Beckenbach, ed., New York, 1961, McGraw-Hill.