Asymptotically optimal declustering schemes for 2-dim range queries

Rakesh K. Sinha\textsuperscript{a,\ast}, Randeep Bhatia\textsuperscript{b}, Chung-Min Chen\textsuperscript{c}

\textsuperscript{a}AT & T Labs.-Research, T D RoomA5-135 200 S. Laurel Avenue, Middletown, NJ 07748, USA
\textsuperscript{b}Bell Labs, Lucent Technologies, USA
\textsuperscript{c}Telcordia Technologies, Inc., USA

Abstract

Declustering techniques have been widely adopted in parallel storage systems (e.g. disk arrays) to speed up bulk retrieval of multidimensional data. A declustering scheme distributes data items among multiple disks, thus enabling parallel data access and reducing query response time. We measure the performance of any declustering scheme as its worst case additive deviation from the ideal scheme. The goal thus is to design declustering schemes with as small an additive error as possible. We describe a number of declustering schemes with additive error $O(\log M)$ for 2-dimensional range queries, where $M$ is the number of disks. These are the first results giving $O(\log M)$ upper bound for all values of $M$. Our second result is a lower bound on the additive error. It is known that except for a few stringent cases, additive error of any 2-dimensional declustering scheme is at least one. We strengthen this lower bound to $\Omega((\log M)^{d-1/2})$ for $d$-dimensional schemes and to $\Omega(\log M)$ for 2-dimensional schemes, thus proving that the 2-dimensional schemes described in this paper are (asymptotically) optimal. These results are obtained by establishing a connection to geometric discrepancy. We also present simulation results to evaluate the performance of these schemes in practice.

\copyright\ 2002 Elsevier Science B.V. All rights reserved.

1. Introduction

In the past decade, the computer manufacturing industry has brought dramatic improvement in CPU power and memory/storage capacity. In contrast, improvement in
disk access time has been relatively flat. As a result, disk I/O is bound to be the bottleneck for many data-intensive scientific applications. To cope with the I/O bottleneck, multi-disk systems, coupled with a declustering scheme, are usually used. The idea is to distribute data blocks across multiple disk devices, so they can be retrieved in parallel, i.e., in parallel disk seek operations. Meanwhile, emerging technologies in Storage Area Networks and fast I/O interconnects (e.g. Fibre Channel, Fast/Wide SCSI, HIPPI-6400, etc.) have also enabled one to build a massively parallel storage system with hundreds or even thousands of disks [25,26,20]. As the number of parallel disks increases, the efficacy of the adopted declustering scheme becomes even more crucial.

Many applications that adopt declustering schemes have to deal with multidimensional data. These applications include, for example, Geographical Information Systems [23] and remote-sensing databases [7,9]. In this paper, we concentrate on two-dimensional (2-dim) data that are organized as uniform grids. Usually, these are raster-spatial or imagery data that can be physically divided into a grid of “tiles”, where each tile is stored as a data block on one of the disks. Remote-sensing data (such as those collected by NASA’s Landsat and France’s SPOT satellites [6,7]) are good examples of raster-spatial data. Ref. [9] provides more details on the physical partition of remote-sensing data into grids of tiles.

An important class of queries associated with multidimensional data is range query. A range query requests a hyper-rectangular subset of the multidimensional data space. The response time of the query is measured by the access time of the disk that has the maximum number of data blocks to retrieve, and our goal is to design declustering schemes that minimize query response time.

In this paper, we measure the performance of any declustering scheme as its worst-case additive deviation from the ideal scheme. Based on this notion, we describe a number of 2-dim schemes with (asymptotically) optimal performance. This is done by giving an upper bound on the performance of each of these schemes as well as a lower bound on the performance of any declustering scheme. These are the first schemes with provably (asymptotically) optimal behavior. Our results are obtained by establishing a connection to geometric discrepancy, a widely studied area of Combinatorics. We have been able to borrow some deep results and machinery from discrepancy theory to prove our results on declustering.

The rest of the paper is organized as follows. First, we formally define the declustering problem. Then, in Section 2, we summarize related work and present a summary of our contributions. In Section 3, we state the intuition behind our results. We briefly describe the relevant results in discrepancy theory and how we use them to prove results on declustering schemes. This is followed in Section 4 by a description of a general technique for constructing good declustering schemes from good discrepancy placements. In Section 6, we describe a number of declustering schemes, all with provably (asymptotically) optimal performance. In Section 7, we present a lower bound argument on the performance of any declustering scheme. Finally, in Section 8, we present brute-force simulation results on 2-dim schemes to show their exact (not asymptotic) performance. The results show that in practice, all the schemes have very good performance: their worst case deviation from the ideal scheme is within 5 for a large range of number of disks (up to 500 disks as the simulation runs so far).
Remark. Our theoretical upper bound holds only for the case of two dimensions. The lower bound (presented in Section 7) applies to any number of dimensions.

1.1. Problem definition

For any positive integer $N$, define $\tilde{N} = \{0, 1, \ldots, N-1\}$. Consider a dataset organized as a $d$-dimensional grid $\tilde{N}_1 \times \tilde{N}_2 \times \cdots \times \tilde{N}_d$, where $N_i$ are integers. Let $(x_1, x_2, \ldots, x_d)$, $x_i \in \tilde{N}_i$, denote a tile (point) in the grid. Given $M$ disks, a declustering scheme $s$ is a function $s : \tilde{N}_1 \times \cdots \times \tilde{N}_d \rightarrow \tilde{M}$, that assigns tile $(x_1, x_2, \ldots, x_d)$ to the disk numbered $s(x_1, x_2, \ldots, x_d)$. A range query $Q$ retrieves a hyper-rectangular set of tiles contained within the grid. Formally, $Q \in I_{N_1} \times I_{N_2} \times \cdots \times I_{N_d}$, where $I_{N_i}$ is the set of all intervals in $\tilde{N}_i$. We define the (nominal) response time of query $Q$ under scheme $s$, $RT(Q, s)$, to be the maximum number of tiles from the query that get assigned to the same disk. Formally, let $tile_i(Q, s)$ represent the number of tiles in $Q$ that get assigned to disk $i$ under scheme $s$. Then

$$RT(Q, s) = \max_{0 \leq i < M} tile_i(Q, s).$$

One may consider the unit of response time to be the average disk access time (including seek, rotational, and transfer time) to retrieve a data block. Thus, the notion of response time indicates the expected I/O delay for answering the query. The problem, therefore, is to devise a declustering scheme that would minimize the query response time.

An ideal declustering scheme would achieve, for each query $Q$, the optimal response time $ORT(Q) = \lceil |Q|/M \rceil$, where $|Q|$ is the number of tiles in $Q$. The additive error of any declustering scheme $s$ is defined as the maximum (over all queries) difference between response time and optimal response time. Formally,

$$\text{additive error of scheme } s = \max_{\forall Q} (RT(Q, s) - ORT(Q)).$$

Note the above definition is independent of grid size (query $Q$ could be as large as possible) and thus the additive error can be unbounded.

The additive error is a measure of the performance of a declustering scheme and thus our goal is to design schemes with the smallest possible additive error.

In the rest of the paper we will not make any distinction between data “tiles” and data “points”. That is we will denote a data tile by a data point. Moreover, for proving our theoretical results, we will frequently omit the ceiling in the expression of the optimal response time. This will change the additive error by at most one.

2. Related work and our contributions

One of the most obvious declustering schemes is to allocate each tile to a randomly chosen disk from the $M$ disks with equal probability. It has been shown through simulation [7] that random assignment results in a poor performance. It is also known (see e.g., [32, p. 52]) that the discrepancy of a random placement of $M$ points in a
unit square is (with high probability) $\sqrt{M \log \log M}$. (Theorems 1 and 2 show a close connection between discrepancy of a placement scheme and the additive error of a declustering scheme.) Intuitively, the reason for the poor performance of randomized scheme is because it fails to take advantage of the structured nature of range queries. In other words, the response time in a random query depends only on the size of the query, rather than on its shape.

Thus schemes that have been specialized for a restricted class of queries (e.g., range queries) tend to perform better in practice. Declustering has been a very well studied problem and a number of such specialized schemes have been proposed [11,13,21,22,8], [35,23,28,9,22,16,4,5,2]. However, very few of these schemes have a good worst case behavior for range queries. (E.g., the 2-dim disk modulo scheme [11] can have additive error as large as $\sqrt{M}$.) We are aware of three schemes with limited guarantee in two-dimensions. These include two of our earlier schemes—GRS scheme [4] and Hierarchical scheme [5]—and a scheme designed by Atallah and Prabhakar [2]. Even these 2-dim guarantees are somewhat weak.

The GRS scheme [4] (described later in this paper) is defined in terms of golden ratio sequences. Although the scheme is defined for all values of $M$, we could prove interesting upper bounds only when $M$ is a Fibonacci number. Specifically, we proved that whenever $M$ is a Fibonacci number the response time of any query is at most three times its optimal response time and the average response time is within 14% of the optimal response time. Using the techniques in this paper, we are now able to prove that this scheme is asymptotically optimal for additive error.

Atallah and Prabhakar [2] proved an additive error of $O(\log M)$ but their scheme and upper bound is defined only when $M$ is a power of two.

The hierarchical scheme [15] is constructed recursively from other base schemes and the resulting performance depends on the performance of these base schemes.

In this paper, for the first time, we prove that a number of 2-dim schemes have additive error $O(\log M)$ for all values of $M$. We also present exhaustive simulation results to show that the exact (not asymptotic) additive error is within 5 for a large range of number of disks. (up to 500 disks as the simulation has run so far). The case of higher (than two) dimensions appear intrinsically very difficult. None of the above mentioned schemes provide any non-trivial theoretical guarantees in higher dimension.

A related question is what is the smallest possible error of a declustering scheme. It is known [35,1] that except for a few stringent cases, additive error of any 2-dim scheme is at least one. We strengthen this lower bound to $\Omega(\log M)$ for 2-dim schemes, thus proving that the 2-dim schemes described in this paper are (asymptotically) optimal. We have also been able to generalize our lower bound to $\Omega((\log M)^{d-1/2})$ for $d$-dim schemes.

These results have been proved by relating the declustering problem to the discrepancy problem—a well studied sub-discipline of Combinatorics. We have borrowed some deep results and machinery from discrepancy theory research to prove our results.

We present a general technique for constructing good declustering schemes from good discrepancy placements. Given that discrepancy theory is an active area of
research, we feel that this may be our most important technical contribution. It leaves open the possibility that one may take new and improved discrepancy placements and translate them into even better declustering schemes. As an evidence of power and generality of our present technique, a straightforward corollary of our main theorem implies a significantly better bound for the GRS scheme than what we had proved in an earlier paper [4].

3. Intuition of our results

All our schemes are motivated by results in discrepancy theory [21]. The fundamental problem of discrepancy theory is: what is the best way to place a set of points such that they are uniformly distributed in a multi-dimensional unit cube. Techniques resulted from the discrepancy theory have found applications in various fields such as numerical integration and geometric searching [24]. In the following, we give a very brief description of the relevant results from discrepancy theory.

3.1. Discrepancy theory

Given any integer $L$, the goal is to determine positions of $L$ points in a two-dim unit square such that these points are placed as uniformly as possible. There are several possible ways of measuring uniformity of any placement scheme. The definition most relevant to us is following:

Fix a placement $P$ of $L$ points and consider any rectangle $R$ that lies completely inside the unit square and whose sides are parallel to the sides of the unit square. If the points were placed completely uniformly in a unit square, we will expect $R$ to contain about $\text{area}(R) \times L$ points, where $\text{area}(R)$ denotes the area of $R$. Measure the absolute difference between $|P \cap R|$ (the actual number of points falling in $R$) and $\text{area}(R) \times L$. This defines the “discrepancy” of placement $P$ with respect to rectangle $R$. The discrepancy of placement $P$ is defined as the highest value of discrepancy with respect to any rectangle $R$. The goal is to design placement schemes with smallest possible discrepancy.

It is known that any placement scheme must have discrepancy at least $\Omega(\log L)$ [30,18] and several placement schemes with discrepancy $O(\log L)$ are known in literature [26]. The definition of discrepancy can be generalized to arbitrary dimensions. In $d$-dimensions, the known lower and upper bounds are $\Omega((\log L)^{(d-1)/2}(\log \log L/ \log \log \log L)^{1/(2d-2)})$ [3,29] and $O((\log L)^{-1})$ [24], respectively. (In the rest of the paper, for simplicity of notation, we will be using the expression $\Omega((\log L)^{(d-1)/2})$ to denote the best lower bound.) These results form the basis of our upper and lower bound arguments.

3.2. Relationship with declustering schemes

We informally argue that a declustering scheme with small additive error can be used to construct a placement scheme with small discrepancy, and vice versa. The
arguments in this subsection are for intuition only. We give precise arguments in the next two sections.

Consider a good declustering scheme on an $M \times M$ grid $G$. Because we are distributing $M^2$ points among $M$ disks, a good declustering scheme will distribute roughly $M$ points to each disk. Let us focus on the $M$ points that got assigned to a specific disk, say disk zero. For any query $Q$, its response time is defined as the maximum number of points assigned to the same disk. We will approximate the response time of $Q$ with the maximum number of points assigned to disk zero and show that this approximate response time is equal to the discrepancy of a placement scheme defined by the geometric positions of these $M$ points.

By approximating the response time of $Q$ with the maximum number of points assigned to disk zero, the additive error of $Q$ becomes equal to

$$\text{tile}_0(Q,s) - \frac{|Q|}{M}.$$ 

Recall that $\text{tile}_0(Q,s)$ denotes the number of instances of disk 0 contained within $Q$.

Now suppose we compress the grid into a unit square (so that both $x$ and $y$ dimensions are compressed by a factor of $M$) and consider the new coordinates of the $M$ points assigned to disk 0. Call these $M$ points placement $P$. Next, consider the original query $Q$ which now gets compressed into a rectangle $R$ of area $|Q|/M^2$. Then, the discrepancy of placement $P$ with respect to $R$ is

$$||P \cap R| - \frac{|Q|}{M^2} * M| \quad \text{(by definition)}$$

$$= \text{tile}_0(Q,s) - \frac{|Q|}{M}.$$ 

Therefore, the discrepancy of the placement scheme is equal to the additive error of the declustering scheme.

The next section describes how to obtain a good declustering scheme from a good placement scheme.

4. From discrepancy to declustering

We will describe our techniques for the case of two dimensions. Our overall strategy can be stated in the following three steps:

1. We start with a (discrepancy) placement scheme $P_0$ in the unit square with $M$ points.
   By multiplying $x$ and $y$ dimensions by $M$, we obtain $M$ points in an $M \times M$ grid. We can think of these points as approximate positions of disk 0. Call this placement $P$.
2. The points in $P$ may not correspond to grid points (i.e., their $x$ or $y$-coordinates may not be integers). In this step we map these $M$ points, obtained in step 1, to $M$ grid points. These grid points determine the tiles in a $M \times M$ grid that are to be assigned to disk 0.
3. In this step, we decide which remaining tiles in the grid should be mapped to disks 1–3, etc. This gives a declustering scheme on an $M \times M$ grid. Finally, we generalize it to an arbitrary $N_x \times N_y$ grid.

The goal is to be able to start with a placement scheme $P_0$ with small discrepancy, and still guarantee a small additive error for the resulting declustering scheme obtained from Steps 1–3 above. Indeed, our construction guarantees that if we start with a placement scheme $P_0$ with discrepancy $k$, then the additive error of the resulting declustering scheme is at most $O(k + k^2/M)$. Thus, picking any placement scheme $P_0$ with $k = O(\log M)$ from the discrepancy theory literature (e.g. [33,14]), we can construct a declustering scheme with $O(\log M)$ additive error. In the rest of this section, we describe Steps 1–3 in detail, along with the necessary claims and their proofs.

4.1. Step 1

Start with a placement scheme $P_0$ that places $M$ points in the unit square $[0,1]^2$ and scale up each dimension by a factor of $M$. Let the resulting points be $(x_0,y_0),(x_1,y_1),$ $(x_2,y_2),\ldots,(x_{M-1},y_{M-1})$. We call this new placement scheme $P$.

Figs. 1 (a) and (b) show an example for $M = 5$. The initial placement $P_0$ is obtained from Faure’s extension of de van Corput’s construction [14]. The resulting placement $P$ in Fig. 1(b) contains points $(0,0),(1.0,2.80),(2.0,1.10),(3.0,3.90)$ and $(4.0,2.20)$.

We will redefine discrepancy for scheme $P$, which is imposed on an $M \times M$ square ([0,$M$]) rather than on a unit square ([0,1]). This new definition differs in two aspects from the standard definition. First of all, we have scaled up the area of any rectangle by a factor of $M^2$ (a scaling of $M$ in each dimension). Second we allow more general rectangles for measuring discrepancy: we consider rectangles whose $x$ or $y$-coordinates may come from a wrap-around interval, i.e., an interval of the form $[a,M-1]\cup[0,b]$, where $0 < b < a < M$ ($a$ and $b$ are real numbers). We denote these as wrap-around rectangles. The introduction of wrap-around rectangles is needed when we generalize the scheme to arbitrary grids in Step 3.
Fig. 2. An example wrap-around rectangle $q_1 \cup q_2 \cup q_3 \cup q_4$.

Fig. 3. Proof of Lemma 2. Points of placement $P$ are denoted by asterisks; points of $P'$ are denoted by solid points. For each point in $P$, its mapping in $P'$ is the nearest dot point. In this example, $|S_{\text{min}}| = 3, |S_\ell| = 5$ and $|S_{\text{max}}| = 6$.

Pictorially, imagine that the left and right sides of the grid are joined and similarly the top and bottom sides of the grid are joined. Because each wrap-around interval is a disjoint union of one or two (standard) intervals, a wrap-around rectangle is a disjoint union of one, two, or four disjoint (standard) rectangles in the grid. The last case will correspond to four standard rectangles in the four corners of the grid. Fig. 2 shows an example.

**Definition 1.** Fix a placement $P$ of $M$ points in the square $[0,M]^2$. Then given any rectangle (which could be a wrap-around rectangle) $R$, the *discrepancy of $P$ with respect to $R$* is defined as

$$\left| \text{number of points falling in } R - M \cdot \frac{\text{area}(R)}{M^2} \right| = \left| |P \cap R| - \frac{\text{area}(R)}{M} \right|.$$ 

The *discrepancy of placement $P$* is defined as the highest value of discrepancy with respect to any (including wrap-around) rectangle $R$.

**Lemma 1.** The discrepancy of $P$ (as defined above) is at most four times the original discrepancy of $P_0$. 


Proof. This can be easily verified by observing that (1) the scaling of $P_0$ to $P$ does not affect the discrepancy value, and (2) the introduction of wrap-around rectangles in $P$ may multiply the discrepancy by at most a factor of four.

4.2. Step 2

From the previous step we obtain a placement of $M$ points $P = \{(x_0, y_0), (x_1, y_1), \ldots, (x_{M-1}, y_{M-1})\}$ on the square $[0, M]^2$. Now impose vertical and horizontal grid lines on the square, spaced at an interval of 1, as shown in Fig. 1(b). For convenience, we will call a vertical (horizontal) grid line a “column” (row). The intersection of a column and a row is called a grid point.

In this step, we will move around the points of $P$ to align with grid points. This is done as follows: place the point in $P$ with the smallest $x$-coordinate in the zeroth column, the point with the next smallest $x$-coordinate in the first column and so on.

We do an analogous thing with $y$-ordinates and rows.

Formally the process is: sort $x_0, x_1, x_2, \ldots, x_{M-1}$ in increasing order (break ties arbitrarily) and let $0 \leq w_i < M$ be the rank of $x_i$ within the sorted order. Similarly sort $y_0, y_1, y_2, \ldots, y_{M-1}$ in increasing order (break ties arbitrarily) and let $0 \leq z_i < M$ be the rank of $y_i$ within the sorted order. Then map $(x_i, y_i)$ to the grid point $(w_i, z_i)$. Let us call the new placement scheme $P'$. Fig. 1(c) shows the resulting $P'$ (in solid points) for the running example.

We will use the following claim in Steps 2 and 3.

Claim 1. Because $w_i$’s are all distinct, each column (among column 0, column 1, \ldots, column $M - 1$) contains exactly one point. Similarly $z_i$’s are all distinct, and thus each row (among row 0, row 1, \ldots, row $M - 1$) contains exactly one point.

Lemma 2. If discrepancy of scheme $P$ is $k$, then the discrepancy of scheme $P'$ is at most $5k + 4 + 4(k + 1)^2/M$. (In all declustering schemes we describe later in Section 6, we will start with an initial placement scheme $P_0$ with discrepancy $O(\log M)$, so that the $4(k + 1)^2/M$ term will be equal to $O(\log M)^2/M$, which is vanishingly small.)

Proof. We will first prove that the positions of point in $P$ do not change too much as we shift them around to obtain $P'$. We will prove that $|w_i - x_i| \leq k + 1$ and $|z_i - y_i| \leq k + 1$.

Consider the rectangle $R$ whose four corners are $(0, 0), (x_i, 0), (0, M)$, and $(x_i, M)$. The area of $R$ is $x_i * M$ and the number of points falling in $R$ is equal to the number of points whose $x$-coordinate is less than or equal to $x_i$, which is $w_i + 1$. So by discrepancy theory, $|w_i + 1 - x_i * M|M| \leq k$, thus $|w_i - x_i| \leq k + 1$. By a similar argument we can prove that $|z_i - y_i| \leq k + 1$. In the following, let $h = k + 1$.

We are ready to prove the lemma. Consider any rectangle $R$ of dimension $c \times r$. Fig. 3 shows the situation. Let $S_R$ denote the set of points from $P'$ that fall inside $R$. We are interested in the cardinality of $S_R$.

Consider the rectangle $R_{\min}$ of dimension $(c - 2h) \times (r - 2h)$ that is obtained by pushing in each side of $R$ by a distance of $h$. Also consider the rectangle $R_{\max}$ of dimension $(c + 2h) \times (r + 2h)$ that is obtained by pushing out each side of $R$ by a
distance of \( h \). Let \( S_{\min} \) (resp. \( S_{\max} \)) denote the set of points from \( P \) that fall inside \( R_{\min} \) (resp. \( R_{\max} \)).

We already showed \( |w_i - x_i| \leq h \) and \( |z_i - y_i| \leq h \). Thus it follows that all the points in \( R_{\min} \) (under scheme \( P \)) must fall in \( R \) (under scheme \( P' \)) and no point outside \( R_{\max} \) (under scheme \( P \)) can fall in \( R \) (under scheme \( P' \)). That is, let \( \psi : P \to P' \) be the one-to-one mapping defined by Step 2. Then, for each point \( x \in S_{\min} \), \( \psi(x) \in S_R \), and for any \( x \in P - S_{\max} \), \( \psi(x) \in P' - S_R \). We conclude

\[
|S_{\min}| \leq |S_R| \leq |S_{\max}|
\]

(1)

Because scheme \( P \) has discrepancy \( k \),

\[
|S_{\min}| \geq \frac{\text{area}(R_{\min})}{M} - k = \frac{(c - 2h)(r - 2h)}{M} - k,
\]

(2)

and

\[
|S_{\max}| \leq \frac{\text{area}(R_{\max})}{M} + k = \frac{(c + 2h)(r + 2h)}{M} + k.
\]

(3)

From Eqs. (1)–(3),

\[
\frac{(c - 2h)(r - 2h)}{M} - k \leq |S_R| \leq \frac{(c + 2h)(r + 2h)}{M} + k.
\]

Discrepancy of \( P' \) with respect to \( R \) is

\[
\left| \frac{|S_R| - c \cdot r}{M} \right| \leq \max \left( \frac{c \cdot r}{M} - \frac{(c - 2h)(r - 2h)}{M} + k, \frac{(c + 2h)(r + 2h)}{M} + k - \frac{c \cdot r}{M} \right)
\]

\[
= \frac{2h(c + r) + 4h^2}{M} + k.
\]

Since \( c, r \leq M \), we get that discrepancy of \( P' \) is bounded by \( 5k + 4 + 4(k + 1)^2/M \).

4.3. Step 3

We will now extend the placement scheme \( P' \) to a declustering scheme on an arbitrary grid. Our strategy can be described as two sub-steps:

**Step 3a:** First, we decluster a \( M \times M \) grid. We map tile \((x, y)\) to disk 0 iff \((x, y) \in P'\). This is shown in Fig. 1(c), where tiles marked with 0 are mapped to disk 0. Then, within any column, if the tile in row \( j \) is mapped to disk 0 then the tile in row \( j + k \mod M, k = 1, \ldots, M - 1 \) is mapped to disk \( k \). We call this declustering scheme \( D \). The result is shown in Fig. 1(d).

**Step 3b:** Replicate the \( M \times M \) pattern obtained from Step 3a to fill any arbitrary grid. That is, map tile \((x, y)\) to the same disk as tile \((x \mod M, y \mod M)\).

Collectively, the two sub-steps can be stated as follows: Sort the points in \( P' \) according to their \( x \)-coordinates. Let the sorted result be \((0, \sigma(0)), \ldots, (M - 1, \sigma(M - 1))\). (Note the sequence \( \sigma = \sigma(0), \ldots, \sigma(M - 1) \) forms a permutation on \( \{0, 1, \ldots, M - 1\} \).)
Then the declustering scheme maps tile \((x, y)\), for arbitrary integers \(x\) and \(y\), to disk \((y - \alpha(x \mod M)) \mod M\).

We need to argue that the resulting declustering scheme has small additive error. This is best done by sub-dividing the proofs into the two corresponding sub-steps.

4.3.1. Proofs for Step 3a

It is easy to see that the mapping \(D\) is well defined because from Claim 1 exactly one tile within any column gets mapped to disk 0. Also every tile in the grid gets mapped to some disk, thus implying this mapping is a declustering scheme.

When computing the additive error of the declustering schemes \(D\), we allow wrap-around range queries, i.e., queries whose \(x\) or \(y\)-coordinates may come from an interval of the form \([j, j+1, \ldots, M - 1, 0, 1, \ldots, k]\), where \(j > k\). This generalized notion is needed for the proofs in Step 3b when we extend the scheme to arbitrary grid size.

We need the following claim for proving Lemma 3.

**Claim 2.** The number of instances of disk 0 in any query \(Q\) is less than or equal to the number of points from \(P'\) falling in \(Q\). That is,

\[
tile_0(Q, D) \leq |P' \cap Q|.
\]

This can be easily verified as if a tile is assigned to disk 0 then its lower-left grid point must be a point in \(P'\), as can be seen from Fig. 1(c).

**Lemma 3.** The additive error of the resulting declustering scheme \(D\) (even w.r.t. wrap-around queries) is less than or equal to the discrepancy of placement \(P'\).

**Proof.** Fix a (possibly wrap-around) range query \(Q\) that gives the highest additive error among all queries. Recall that the additive error is defined as the difference in response time \(RT(Q)\) and the optimal response time \(|Q|/M\). (For notational convenience, we will ignore the ceiling in the expression of the optimal response time. This can only overestimate the additive error.) \(RT(Q)\), in turn, is defined as the maximum number of instances of any disk within \(Q\). Find a disk number (say \(j\)) such that \(Q\) contains \(RT(Q)\) instances of disk \(j\).

Find another (possibly wrap-around) query \(Q'\) that is obtained by shifting \(Q\) \(j\) positions down. Formally, point \((x, y - j \mod M)\) is in \(Q'\) iff point \((x, y)\) is in \(Q\). By our Step 3a construction, the number of instances of disk 0 in \(Q'\) is equal to number of instances of disk \(j\) in \(Q\). Thus, the additive error of \(Q\) under \(D\)

\[
= RT(Q) - \frac{|Q|}{M}
\]

\[
= \text{number of instances of disk } j \text{ in } Q - \frac{|Q|}{M}
\]

\[
= \text{number of instances of disk } 0 \text{ in } Q' - \frac{|Q'|}{M}
\]
Fig. 4. Let \( AR(Q) \) denote additive error of \( Q \). Then \( AR(Q) = AR(Q') = AR(q_1 \cup q_2) \).

\[
\leq |P' \cap Q'| - \frac{|Q'|}{M} \quad \text{(by Claim 2)}
\]

\[
= \text{discrepancy of } P' \text{ w.r.t. } Q' \quad \text{(by Definition 1)}
\]

\[
\leq \text{discrepancy of } P'. \quad \square
\]

4.3.2. Proofs for Step 3b

Let \( D' \) be the extended declustering scheme on an arbitrary grid obtained in Step 3b.

**Lemma 4.** The additive error of the resulting declustering scheme \( D' \) is equal to the additive error of the declustering scheme \( D \) in Step 3a.

**Proof.** From Claim 1 and the constructions in Step 2 and 3, we conclude that scheme \( D' \) is a permutation scheme, that is, each row and column of size \( M \) in the grid is a permutation of \( \{0, 1, 2, \ldots, M - 1\} \). We show formally in [5] that if the declustering scheme is a permutation scheme then given any range query \( Q \), its additive error is equal to the additive error of a (possibly wrap-around) query in the left-bottom \( M \times M \) grid, as shown in Fig. 4. The intuition of the proof is that we can continue “chopping off” any blocks of size, \( M \times 1 \) and \( 1 \times M \) from \( Q \) until we are left with a query \( Q' \) with both dimensions less than \( M \), and the additive error of \( Q \) will be the same as that of \( Q' \). The reason is that because we have a permutation scheme, any block of size \( M \times 1 \) or \( 1 \times M \) contains exactly one instance of each disk. Thus it contributes exactly one to the response time. Since the area of this block is \( M \), it contributes one to the optimal response time as well. Therefore, removing this block does not change the additive error (defined as the difference of response time and optimal response time). We can project \( Q' \) to the left-bottom \( M \times M \) grid to obtain a (possibly wrap-around) query of the same additive error.

Lemmas 1, 2, 3, and 4 immediately give our main upper bound theorem.

**Theorem 1.** If we start with a placement scheme \( P_0 \) with discrepancy \( k \) then the resulting declustering scheme \( D' \) will have additive error at most \( O(k + k^2/M) \). This
implies that by starting with a placement scheme with \(O(\log M)\) discrepancy, we can construct a declustering scheme with \(O(\log M)\) additive error.

In Section 7, we will prove a converse of this theorem (Theorem 2). In Section 8, we present exhaustive simulation results to show that the exact (not asymptotic) additive error is quite small for a large range of number of disks.

5. Higher-dimensional extensions

In the previous section, we described a general technique for constructing good 2-dim declustering schemes from good discrepancy placements. There are several possible ways of extending this technique to obtain higher-dim declustering schemes.

We can start with a higher-dimensional placement scheme with good discrepancy to obtain a higher-dimensional declustering scheme. Alternatively, we can start with a 2-dim declustering scheme (obtain using the technique outlined in the previous section) and generalize it to higher dimension as follows:

Let \(E_{ESC}\) be the permutation in Step 3 of the previous section. Then in \(d\)-dimensions, the point \((x_1, x_2, \ldots, x_d)\) is mapped to the disk (with all operations modulo \(M\)) \(x_d - \sigma(x_{d-1} - \sigma(x_{d-2} - \cdots))\).

Please note that this is a generalization of the 2-dim declustering scheme which maps point \((x_1, x_2)\) to disk \((x_2 - \sigma(x_1))\) (with all operations modulo \(M\)). The intuition for this generalization is as follows. We want a declustering scheme in which two points which are mapped to the same disk are either “well” declustered in the \(d\)th dimension or are well declustered when projected onto the first \(d - 1\) dimensions. This suggests a recursive construction of the declustering scheme. The details of these constructions and their evaluations appear in [10].

6. Declustering schemes based on low-discrepancy placements

In this section, we present several 2-dim declustering schemes with \(O(\log M)\) additive errors. They are constructed based on placement schemes with \(O(\log M)\) discrepancy, according to the steps described in Section 4. We describe more than one schemes in the hope that users may have other constraints (besides trying to minimize response time) and some of these schemes may be better suited than the others. For comparison purpose, we include another earlier declustering scheme—the Hierarchical scheme [15], which also has a \(O(\log M)\) additive errors. The actual performance of these schemes are compared in Section 8.

6.1. Corput’s scheme

The first placement scheme we consider is given by Van der Corput [33,34]. The \(M\) points given by this scheme are

\[
\left\{ \left( \frac{i}{M}, r_i \right), 0 \leq i < M \right\},
\]
where \( r_i \) is computed as following: let \( a_{k-1} \ldots a_1a_0 \) be the binary representation of \( i \), where \( a_0 \) is the least significant bit. That is, \( i = a_0 + 2a_1 + 2^2a_2 + 2^3a_3 + \ldots \). Then \( r_i = a_0/2 + a_1/4 + a_2/8 + \ldots + a_{k-1}/2^k \).

Now we apply the three steps outlined in the previous section. In Step 1, we multiply each coordinate by \( M \) to obtain the set \( \{(i,M \times r_i), 0 \leq i < M\} \). In Step 2, we need to map these points to integer co-ordinates. The \( x \)-coordinate is already an integer. The \( y \)-coordinate \( M \times r_i \) gets mapped to the rank of \( M \times r_i \) in the set \( \{M \times r_i, 0 \leq i < M\} \).

We observe that this is equal to the rank of \( r_i \) in the set \( \{r_i, 0 \leq i < M\} \). Let \( \text{RANK}(i) \) denote the rank of this element. Then Step 3 dictates that the point \((x,y)\) should map to disk \((y - \text{RANK}(x \mod M)) \mod M\).

We summarize these steps below.

**Step 1:** Construct \( M \) pairs \((i, ri)\) for \( 0 \leq i < M \), where \( ri \) is computed as following: Let \( a_{k-1} \ldots a_1a_0 \) be the binary representation of \( i \), where \( a_0 \) is the least significant bit. Then \( r_i = a_0/2 + a_1/4 + a_2/8 + \ldots + a_{k-1}/2^k \).

**Step 2:** Sort the first components based on the values of \( ri \). This will give a permutation on \( 0,1,\ldots,M-1 \). Call the resulting permutation \( \text{PERM}(M) \). Compute the inverse permutation, \( \text{RANK} \) by

\[
\text{for } i = 0 \text{ to } M - 1 \{ \text{RANK}(\text{PERM}(i)) = i \}
\]

**Step 3:** map point \((x,y)\) to disk \((y - \text{RANK}(x \mod M)) \mod M\).

The next two declustering schemes are constructed in an analogous manner, except that they start with a different initial placement. Rather than describing the steps from discrepancy to declustering scheme, we simply present the initial placement schemes.

### 6.2. Golden ratio sequence (GRS) scheme

The GRS scheme was first described in [4]. In the paper, we proved that whenever \( M \) is a Fibonacci number, the response time of any query is at most three times its optimal response time. We also proved that the GRS scheme has a very good average case behavior whenever \( M \) is a Fibonacci number. Our proof was in terms of “gaps” of any permutation and worked only when \( M \) was a Fibonacci number.

It turns out that the same scheme can be obtained from a placement scheme with discrepancy \( O(\log M) \) [19] [24, p. 80, Exercise 3] described below. Thus Theorem 1 implies that the additive error of GRS scheme is \( O(\log M) \) for any \( M \). This is another evidence of the generality and power of Theorem 1.

The placement scheme that corresponds to the GRS declustering scheme belongs to a class of low-discrepancy schemes called lattice sets [24]. The \( M \) points generated are:

\[
\left\{ \left( \frac{i}{M}, \left\{ \frac{2i}{1 + \sqrt{5}} \right\} \right) \right\}, \ 0 \leq i < M,
\]

where \( \{x\} \) stands for the fractional part of \( x \).
6.3. Faure’s scheme

This declustering scheme is based on Faure’s placement scheme [15]. This scheme has two parameters: a base $b$ and a permutation $\sigma$ on $\{0, 1, \ldots, b - 1\}$. For suitable choice of the parameters, this is the best known construction (in terms of the constant factor in the discrepancy bound).

The $M$ points generated are:

\[ \left\{ \left( \frac{i}{M}, r_i \right) : 0 \leq i < M \right\}, \]

where $r_i$ is defined as follows. Let $a_k \ldots a_1 a_0$ be the representation of $i$ in base $b$, where $a_0$ is the least significant digit. Then

\[ r_i = \sigma(a_0)/b + \sigma(a_1)/b^2 + \sigma(a_2)/b^3 + \cdots + \sigma(a_{k-1})/(b^k). \]

Note that the Corput scheme is a special case with $b = 2$ and $\sigma$ being the identity permutation.

6.4. Net-based scheme

A large class of schemes with low discrepancy are based on $b$-ary nets [27,31,14,24]. In general, these schemes are defined only when $M$ is a power of a prime number. Here we give an example of a declustering scheme based on one such $b$-ary net discrepancy scheme (Faure’s construction [14]). We describe the scheme first (which is defined only when $M$ is a prime power) and then present a technique to generalize the scheme to all values of $M$, with $O(\log M)$ additive errors (proof omitted).

In the following we will assume $M = b^n$, $m \geq 2$. Let $C_0, C_1$ be two $m \times m$ matrices, with entries in the range $0, \ldots, b - 1$, where $C_0$ is the identity matrix and the entry $a_{(i,j)}$ of $C_1$ is 0 if $j < i$ and $a_{(i,j)} = \left( i_j - i_{j+1} \right)$ where all the arithmetic is in the field $GF(b)$. Let $v_0, v_1 \in GF(b)^m$ be two $m$-dimensional vectors, each initialized to the 0 vector. Let $B = (b^{-1}, b^{-2}, \ldots, b^{-n})$.

Step 1: Construct $M = b^n$ pairs $(x_i, y_i)$ for $0 \leq i < M$ in the following loop:

Set $x_i = v_0 \cdot B$ (vector dot product) and $y_i = v_1 \cdot B$.

Let $i = a_1 + ba_2 + \ldots$ be the $b$-ary representation of $i$,

and let $j$ be the smallest index such that $a_j \neq b - 1$.

Add the $j$th column of $C_0$ to $v_0$ and add the $j$th column of $C_1$ to $v_2$.

Step 2: Define function $NET_b(x_i) = y_i$.

Step 3: Map point $(x, y)$ to disk $(y - NET_b(x \mod M)) \mod M$.

The fact that $NET_b$ is a well defined function follows from the fact that in this construction $x_i$ takes on all possible values in the range $0 \ldots M - 1$ [14,24].
A hashing trick

The net-based scheme described above is defined only when $M$ is a prime number. Here we describe a hashing trick that extends the scheme to all values of $M$. This technique is a general technique that can be applied to any declustering scheme that satisfies the following conditions: (1) the scheme must be a column permutation declustering scheme (CPDS), and (2) for any $M = n$ where the scheme is not defined, there must exist an $M = n'$, $n' > n$, such that the scheme is defined.

A CPDS scheme is defined by a permutation $\sigma = \sigma(0), \ldots, \sigma(M - 1)$ on $\{0, 1, \ldots, M - 1\}$, such that the grid point $(x, \sigma(x))$, $x = 0, 1, \ldots, M - 1$, is assigned to disk 0 and, in general, point $(x, y)$ is assigned to disk $(y - \sigma(x \mod M)) \mod M$. We have seen such schemes earlier: for example, the declustering scheme $D$ in Fig. 1(d) is a CPDS. Given a CPDS which is only defined for certain values of $M$, we extend it to all values of $M$ as follows: given $M = n$ for which the CPDS is not defined, we find the smallest number $n' \geq n$, such that the CPDS is defined for $M = n'$. Consider an $n' \times n'$ grid $G$ under the CPDS. Again, we will restrict our attention to disk 0 in this grid. We construct a CPDS for $M = n$ as follows. Take the first $n$ columns of $G$. Note that there are $n$ instances of disk 0 in these columns. We assign each of these disks a unique rank (essentially sort them, break ties in any way) between 0 and $n' - 1$, based on their row positions ($y$-coordinates). Let the rank of disk 0 in column $i$ be $RANK(i)$. Define a permutation $\sigma(i) = RANK(i)$. The CPDS defined by permutation $\sigma$ is the CPDS for $M = n$.

6.5. Generalized hierarchical scheme

The Hierarchical Scheme [5] is based on a technique of constructing declustering scheme for $M = m_1 \times m_2 \times \cdots \times m_k$ disks, given declustering schemes $D_i$ for $m_i$, $1 \leq i \leq k$, disks. Note that $m_i$ may be the same as $m_j$ for $i \neq j$. The idea is that using nearly optimal declustering schemes for small number of disks, one can construct a good declustering scheme for any number (of disks) that has small prime factors. As an example, we may start with good declustering schemes for the first $p$ prime numbers. In this case, the hierarchically constructed declustering schemes are only defined for those $M$ which can be expressed as a product of the first $p$ prime numbers. We refer the readers to [5] for the detailed description of hierarchical schemes.

The original hierarchical declustering scheme is a column permutation declustering scheme (CPDS), as defined before. Hence as shown before we can use the hashing technique defined earlier to create a declustering scheme which works for all $M$. We call this the generalized hierarchical declustering scheme. We can show that if we start with optimal base schemes, then the resultant generalized hierarchical declustering scheme has a $O(\log M)$ additive error (proof omitted).

7. Lower bound

An ideal declustering scheme $s$ is a scheme whose performance is optimal on all range queries, i.e., $RT(Q, s) = |Q|/M$ for all $Q$. An interesting question is whether any
realizable declustering scheme is ideal, and if not how close can any declustering scheme get to the ideal scheme. It is known [35,2] that except for a few stringent cases, additive error of any 2-dim scheme is at least one. We strengthen the result to give (asymptotically) tight lower bound of \( \Omega(\log M) \) in 2-dim and a lower bound of \( \Omega((\log M)^{(d-1)/2}) \) for any \( d \)-dim scheme, thus proving that the 2-dim schemes described in this paper are (asymptotically) optimal.

**Theorem 2.** Let \( LB(d) \) be a lower bound on the discrepancy of any \( d \)-dimensional placement scheme. Then given any \( d \)-dim declustering scheme for \( M \) disks and any \( M \times M \times \cdots \times M \) grid \( G \), there exists a query \( Q \) in the grid \( G \) with \( RT(Q) - ORT(Q) = \Omega(\log M) \). In other words, for any \( d \)-dim declustering scheme, there are queries on which the response time is at least \( \Omega(LB(d)) \) more than the optimal response time. The constant in the Omega expression depends on the number of dimensions, \( d \).

Based on the known lower bounds on discrepancy [30,18,3,29] (see Section 3 for summary), we obtain the following corollaries.

**Corollary 1.** Given any 2-dim declustering scheme \( D \) for \( M \) disks and any \( M \times M \) grid \( G \), there exists a query \( Q \) in the grid \( G \) with \( RT(Q) - ORT(Q) = \Omega(\log M) \). In other words, for any 2-dim declustering scheme, there are queries on which the response time is at least \( \Omega(\log M) \) more than the optimal response time.

**Corollary 2.** Given any \( d \)-dim declustering scheme \( D \) for \( M \) disks and any \( M \times M \times \cdots \times M \) grid \( G \), there exists a query \( Q \) in the grid \( G \) with \( RT(Q) - ORT(Q) = \Omega((\log M)^{(d-1)/2}) \). In other words, for any \( d \)-dim declustering scheme, there are queries on which the response time is at least \( \Omega((\log M)^{(d-1)/2}) \) more than the optimal response time. The constant in the Omega expression depends on the number of dimensions, \( d \).

To keep the notation simple, we will first prove the theorem for \( d = 2 \). Then we will describe the changes needed to generalize the proof to the case of larger \( d \).

**Proof.** Let \( D \) be a declustering scheme and \( G \) be an \( M \times M \) grid as in the statement of the theorem. Since the \( M^2 \) grid points in \( G \) are mapped to the disks \( \{0 \ldots M - 1\} \), there must exist a disk \( i \in \{0 \ldots M - 1\} \), such that there are at least \( n \geq M \) instances of disk \( i \) in \( G \). Without loss of generality, we assume \( i = 0 \). Let us remove all disks except disk 0 from \( G \). Let us also remove \( n - M \) instances of disk 0 from \( G \), thus leaving exactly \( M \) points (disks) in \( G \). We will denote by \( p(Q) \) the number of points contained in a rectangular query \( Q \). We will show that there is a query \( Q \) such that \( p(Q) - ORT(Q) = \Omega(LB(2)) \). Because there are at least \( p(Q) \) instances of disk 0 in \( Q \) under the declustering scheme \( D \), this will imply \( RT(Q) - ORT(Q) \geq p(Q) - ORT(Q) = \Omega(LB(2)) \).

Our proof strategy is following: We will obtain a placement scheme from the positions of the \( M \) points in \( G \). It is known that any placement scheme has discrepancy \( LB(2) \) with respect to at least one rectangle \( R \). We want to use \( R \) to construct a query
Fig. 5. At least one of the nine rectangles has a large positive discrepancy.

Q with $\Omega(LB(2))$ additive error. There are two problems with this simple plan. The first problem is that the boundary of $R$ may not be aligned with the grid lines. This is fixed by taking a slightly smaller rectangle whose boundary lies on the grid lines and arguing that this new rectangle also has a high discrepancy. The second problem is more serious. Remember that discrepancy is defined as the absolute difference of expected and actual number of points. So it is possible that $R$ may be receiving fewer points than expected. In this case, we can only claim that the corresponding query is receiving fewer (than ORT) instances of disk 0, not enough to prove a large additive error. The way around is to observe that the grid $G$ as a whole has zero discrepancy. So if $R$ receives fewer points than expected, some other rectangle must be receiving more points than expected, and we construct a query from that rectangle. The details of the proofs follow:

Let us scale all distances by $1/M$, thus giving us a unit square with $M$ points in it (corresponding to the positions of the disk 0 in $G$). Because of our assumption on the lower bound of the discrepancy of any placement scheme, we have a rectangle $Q'$, in the unit square, containing $p(Q')$ points, for which we have $|M\ Area(Q') - p(Q')| \geq LB(2)$. The first step in the proof is to show the existence of a rectangle $Q''$ with large positive discrepancy.

There are two cases: either for rectangle $Q'$ we have $p(Q') - M\ Area(Q') \geq LB(2)$, in this case we set $Q'' = Q'$. Or for rectangle $Q'$ we have $M\ Area(Q') - p(Q') \geq LB(2)$, in which case we show that there is another rectangle $Q''$ with large (positive) discrepancy. In order to show this, we partition the grid $G$ into at most 9 queries with $Q_1 = Q'$ as shown in Fig. 5. We claim that for one of these queries $Q_i$, we have $p(Q_i) - M\ Area(Q_i) = \Omega(LB(2))$. This is shown as follows. Because we left exactly $M$ points in the grid $G$, $\sum_{i=1}^{9} p(Q_i) = M$. Also, because of the unit area we have, $\sum_{i=1}^{9} Area(Q_i) = 1$. Thus

$$\sum_{i=1}^{9} (p(Q_i) - M\ Area(Q_i)) = \left(\sum_{i=1}^{9} p(Q_i)\right) - \left(\sum_{i=1}^{9} M\ Area(Q_i)\right) = M - M = 0.$$
Fig. 6. Q* is the largest rectangle contained in Q'' that lie on grid lines.

But by assumption \( p(Q_i) - M \text{Area}(Q_i) \leq -LB(2) \). Hence by combining we get

\[
\sum_{i=2}^{9} (p(Q_i) - M \text{Area}(Q_i)) \geq LB(2).
\]

Thus we must have for some \( Q_i \) that \( p(Q_i) - M \text{Area}(Q_i) \geq LB(2)/8 \). We set \( Q'' \) to be this particular \( Q_i \). Next we obtain a boundary-aligned rectangle \( Q^* \) with positive discrepancy.

Let \( Q^* \) be another rectangle contained in \( Q'' \) such that \( Q^* \) is the largest (with maximum area) rectangle contained in \( Q'' \) whose boundary lies on the grid lines of \( G \) in the unit square. Note that \( Q^* \) contains \( p(Q^*) = p(Q'') \) points. This is because all the \( p(Q'') \) points in the rectangle \( Q'' \) lie on the grid lines of \( G \) in the unit square. An example of these queries is shown in Fig. 6. Hence we have

\[
p(Q^*) - M \text{Area}(Q^*) \geq p(Q'') - M \text{Area}(Q'') = \Omega(LB(2)).
\]

Hence we have a rectangle \( Q^* \), containing \( p(Q^*) \) points, in the unit square whose boundary lies on the grid lines of \( G \) in the unit square, such that \( p(Q^*) - M \text{Area}(Q^*) = \Omega(LB(2)) \). Let us scale back all distances by \( M \). Note that the rectangle \( Q^* \) in the unit square becomes a range query \( Q \) in the \( M \times M \) grid \( G \), such that \( Q \) contains \( p(Q) = p(Q^*) \) points and \( |Q| = M^2 \text{Area}(Q^*) \). Thus for query \( Q \) we have

\[
p(Q) - \text{ORT}(Q) = p(Q) - \lceil |Q|/M \rceil = p(Q^*) - \lceil M \text{Area}(Q^*) \rceil = \Omega(LB(2)).
\]

We claim that for the declustering scheme \( D \) and for the query \( Q \), \( RT(Q) - \text{ORT}(Q) = \Omega(LB(2)) \). This follows since there are at least \( p(Q) \) instances of disk 0 in \( Q \) under the declustering scheme \( D \). Hence for query \( Q \) we have \( RT \geq p(Q) \). Thus \( RT(Q) - \text{ORT}(Q) \geq p(Q) - \text{ORT}(Q) = \Omega(LB(2)) \).

The only change needed to generalize the proof to the case of higher dimensions is: in the first part, instead of partitioning the grid \( G \) into 9 queries, we partition it into \( 3^d \) queries. \( \Box \)
8. Simulation results

First, we present simulation results that compare the actual additive errors of the various 2-dim schemes described in the previous section. For Faure’s scheme, we tried three variations: $b = 5, \sigma = 0, 3, 2, 1, 4$; $b = 9, \sigma = 0, 5, 2, 7, 4, 1, 6, 3, 8$; and $b = 36, \sigma = 0, 25, 17, 7, 31, 11, 20, 3, 27, 13, 34, 22, 5, 15, 29, 9, 23, 1, 18, 32, 8, 28, 14, 4, 21, 33, 12, 26, 2, 19, 10, 30, 6, 16, 24, 35$. For the generalized hierarchical scheme, it is constructed solely based on the three optimal schemes for $M = 2, 3$ and 5.

We also include a random scheme for comparison. We use a random-assignment scheme that guarantees a balanced distribution of tiles among all disks. It works as follows: First, label the tiles in the grid in row-major order as $0, 1, \ldots, N - 1$, where $N = N_xN_y$. Generate a random permutation of the numbers, say $a_0, a_1, \ldots, a_{N - 1}$. Assign the tile labeled $a_i$ to disk $i \mod M$.

We compute the additive errors of each scheme for all values of $M$ (in a range). For a fixed $M$, we run exhaustive simulation to compute exact worst additive error. As argued in the proof of Lemma 4, in order to compute the additive error of any permutation declustering scheme, it is enough to consider all possible queries, including wrap-around queries, in an $M \times M$ grid. We vary $M$ from two to a few hundreds. Because the simulation is time consuming, the results we present here are what we have obtained at the time of the writing. For each scheme, the running time for the maximum value of $M$ has exceeded 24 h.

We present the results in Table 1. The numbers at the top row represent additive errors. The number in each of the table entries represents the maximum number of disks for which the corresponding additive error is guaranteed. For example, Corput’s scheme guarantees that when $M \leq 3$ the additive error is 0; when $M \leq 8$ its additive error is at

<table>
<thead>
<tr>
<th>additive error =</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>39</th>
</tr>
</thead>
<tbody>
<tr>
<td>Corput</td>
<td>3</td>
<td>8</td>
<td>34</td>
<td>130</td>
<td>273</td>
<td>470</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Faure-b5</td>
<td>3</td>
<td>10</td>
<td>27</td>
<td>106</td>
<td>140</td>
<td>275</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Faure-b9</td>
<td>3</td>
<td>9</td>
<td>45</td>
<td>90</td>
<td>338</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Faure-b36</td>
<td>3</td>
<td>11</td>
<td>36</td>
<td>109</td>
<td>306</td>
<td>368</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>GRS</td>
<td>3</td>
<td>22</td>
<td>94</td>
<td>391</td>
<td>519</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>FaureNet</td>
<td>—</td>
<td>4</td>
<td>9</td>
<td>49</td>
<td>49</td>
<td>49</td>
<td>125</td>
<td>—</td>
</tr>
<tr>
<td>FaureNet-H</td>
<td>3</td>
<td>7</td>
<td>15</td>
<td>45</td>
<td>49</td>
<td>57</td>
<td>125</td>
<td>—</td>
</tr>
<tr>
<td>Hierarchical</td>
<td>3</td>
<td>10</td>
<td>45</td>
<td>90</td>
<td>200</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Hierarchical-H</td>
<td>3</td>
<td>11</td>
<td>45</td>
<td>95</td>
<td>200</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Random</td>
<td>—</td>
<td>—</td>
<td>—</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>67</td>
</tr>
</tbody>
</table>

1 Given $M$, it takes $O(M^6)$ to compute the additive error of a scheme. For some schemes, we manage to reduce the complexity to $O(M^5)$ or $O(M^4)$ by taking advantage of a dynamic programming technique and some special properties of the schemes.
Table 2
Comparison of average query response time. Grid size = 30 × 31

<table>
<thead>
<tr>
<th>Number of disks</th>
<th>8</th>
<th>21</th>
<th>41</th>
<th>60</th>
<th>130</th>
<th>230</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>18.98</td>
<td>9.42</td>
<td>6.08</td>
<td>4.81</td>
<td>3.31</td>
<td>2.57</td>
</tr>
<tr>
<td>Corput</td>
<td>15.15</td>
<td>6.38</td>
<td>3.87</td>
<td>2.90</td>
<td>1.84</td>
<td>1.41</td>
</tr>
<tr>
<td>Faure-b9</td>
<td>15.12</td>
<td>6.38</td>
<td>3.84</td>
<td>2.94</td>
<td>1.86</td>
<td>1.43</td>
</tr>
<tr>
<td>GRS</td>
<td>15.12</td>
<td>6.25</td>
<td>3.76</td>
<td>2.83</td>
<td>1.78</td>
<td>1.37</td>
</tr>
<tr>
<td>FaureNet-H</td>
<td>15.28</td>
<td>6.38</td>
<td>4.09</td>
<td>3.65</td>
<td>1.84</td>
<td>1.76</td>
</tr>
<tr>
<td>Hierarchical-H</td>
<td>15.11</td>
<td>6.34</td>
<td>3.86</td>
<td>2.88</td>
<td>1.89</td>
<td>1.47</td>
</tr>
</tbody>
</table>

most 1; when $M \leq 34$, the additive error is at most 2, etc. The row labeled FaureNet represents the scheme based on Faure’s net construction, where $M$ is restricted to prime powers (i.e. $M = b^m$, where $b$ is a prime and $m \geq 2$.) The row labeled FaureNet-H represents the one that has been generalized to all values of $M$ using our proposed hashing trick. The same stands for Hierarchical and Hierarchical-H, where the former restricts $M$ to products of powers of 2, 3 and 5 and the latter is applicable to all values of $M$.

The table shows that all schemes provide much better worst case performance than the random scheme. For all schemes (except the random scheme), the additive deviations are at most 5 for a large range of $M$. The most notable is GRS, which guarantees an additive error at most 4 for $M \leq 519$. The net-based schemes, FaureNet and FaureNet-H, turn out to be the least efficient ones among all. We found that when $M$ is a large power of 2, FaureNet gives higher additive errors than usual. For example, its additive error is 6 when $M = 64$, whereas it is 3 for the preceding prime power ($M = 49$) and 4 for the successive prime power ($M = 81$). The same scenario appears at $M = 128$, which has an additive error of 10, whereas the additive error is 6 for both the preceding ($M = 125$) and successive ($M = 169$) prime powers.

The hashing trick we proposed to extend the Hierarchical and FaureNet schemes to all values of $M$ also turn out to work well in practice. For example, when extending the original Hierarchical scheme (with restricted $M$) to the generalized Hierarchical-H scheme (which applies to all values of $M$), the additive errors are largely preserved, as can be observed in the respective rows in the table.

The next experiment compares the schemes in terms of average query response time. We fix a grid size and compare the average query response time of the schemes under different numbers of disks. The average is computed by considering all possible range queries within the grid. We choose a grid size of $30 \times 31$, which is based on the parameters of a real-world remote-sensing database [17]. In that database, each “Scene” image from the NASA Landsat 5 satellite [6] is represented as a grid of $30 \times 31$ tiles, where each tile contains $191 \times 210$ pixels = 40,110 bytes.

We vary the number of disks from 8, 21, 41, 60, 130 to 230. Table 2 shows the average response time of the schemes. Among all, GRS gives the best performance, while all discrepancy-based schemes outperform the random scheme. Fig. 7 depicts the improvement (in percentage) of the discrepancy-based schemes over the random scheme. As the number of disks increases, the improvement (over random-assignment)
of all schemes, except FaureNet-H, increases. The improvement, however, levels off when the number of disks exceeds a certain threshold (in this case 130). Eventually, the improvement should begin to decline as the number of disks grows towards the grid size (in which case the query response time will become so small and leave little room for improvement over a random scheme). The sudden performance drop of Faurenet-H at $M = 60$ reflects the fact that it is constructed (using the hashing trick) from the next prime power $M = 64$. And we already know that FaureNet gives not-so-good performance when $M$ is a large power of 2.

Finally, we note that it should not be taken that the GRS scheme is better than all other schemes for all values of $M$. A better strategy is to use a hybrid scheme: given $M$, select the scheme with the lowest additive error.

Given the small additive errors of these schemes, we feel that other performance metrics, such as average additive error and ratio to the optimal, are of less importance. An additive error within 5 translates into less than 50m difference in practice (assuming 10 ms disk seek time). Taking into account seek time variation, the response time is already optimal in a statistical sense. Nonetheless, we leave it to the users to select from these schemes the one that best fits their requirements (for example in a multiuser environment the average response time may be more important).

9. Conclusions

Declustering is a popular technique to speed up bulk retrieval of multidimensional data. This paper focuses on range queries for uniform data. Even though this is a very well-studied problem, none of the earlier proposed schemes have provable good
behavior. We measure the additive error of any declustering scheme as its worst case additive deviation from the ideal scheme. In this paper, for the first time, we describe a number of 2-dim schemes with additive error $O(\log M)$ for all values of $M$. We describe higher-dimensional generalization of these declustering schemes. We also present brute-force simulation results to show that the exact (not asymptotic) additive error is quite small for a large range of number of disks. We prove that this is the best possible analytical bound by giving a matching lower bound of $\Omega(\log M)$ on the performance of any 2-dim declustering scheme. We generalize this lower bound to $\Omega((\log M)^{(d-1)/2})$ for $d$-dimensional schemes.

Our main technical contribution is a connection between declustering problem and discrepancy theory, a well studied sub-discipline of Combinatorics. We give a general technique for mapping any good discrepancy placement scheme into a good declustering scheme. As an evidence of power and generality of our present technique, a straightforward corollary of our main theorem implies a significantly better bound for the GRS scheme than what we were able to prove in an earlier paper [4].

Acknowledgements

We are indebted to Jirka Matousek for his wonderful book on discrepancy theory as well as his prompt responses to all our questions.

References