Adaptive Succinctness*

Diego Arroyuelo[1,2] and Rajeev Raman[3]

[1] Millennium Institute for Foundational Research on Data (IMFD), Chile.
[2] Department of Informatics, Universidad Técnica Federico Santa María, Chile.
darroyue@inf.utfsm.cl
[3] Department of Informatics, University of Leicester
University Road, Leicester, LE1 7RH, United Kingdom.
r.raman@leicester.ac.uk

Abstract. Representing a static set of integers S, $|S| = n$ from a finite universe $U = [1..u]$ is a fundamental task in computer science. Our concern is to represent S in small space while supporting the operations of rank and select on S; if S is viewed as its characteristic vector, the problem becomes that of representing a bit-vector, which is arguably the most fundamental building block of succinct data structures.
Although there is an information-theoretic lower bound of $\mathcal{B}(n,u) = \lg \binom{u}{n}$ bits on the space needed to represent S, this applies to worst-case (random) sets S, and sets found in practical applications are compressible. We focus on the case where elements of S contain non-trivial *runs* of consecutive elements, one that occurs in many practical situations.
Let $\mathcal{C}_n$ denote the class of $\binom{u}{n}$ distinct sets of n elements over the universe $[1..u]$. Let also $\mathcal{C}_g^n \subset \mathcal{C}_n$ contain the sets whose n elements are arranged in $g \leq n$ runs of $\ell_i \geq 1$ consecutive elements from U for $i = 1, \ldots, g$, and let $\mathcal{C}_{g,r}^n \subset \mathcal{C}_g^n$ contain all sets that consist of g runs, such that $r \leq g$ of them have at least 2 elements.

- We introduce new compressibility measures for sets, including:
  - $\mathcal{L}_1 = \lg |\mathcal{C}_g^n| = \lg \binom{u-n+1}{g} + \lg \binom{n-1}{g-1}$ and
  - $\mathcal{L}_2 = \lg |\mathcal{C}_{g,r}^n| = \lg \binom{u-n+1}{g} + \lg \binom{n-g-1}{r-1} + \lg \binom{g}{r}$

  We show that $\mathcal{L}_2 \leq \mathcal{L}_1 \leq \mathcal{B}(n,u)$.
- We give data structures that use space close to bounds $\mathcal{L}_1$ and $\mathcal{L}_2$ and support rank and select in $O(1)$ time.
- We provide additional measures involving entropy-coding run lengths and gaps between items, data structures to support these measures, and show experimentally that these approaches are promising for real-world datasets.

1 Introduction

Given a static sorted set $S = \{x_1, \ldots, x_n\}$ of n elements from a finite universe $U = [1..u] \subset \mathbb{N}$, such that $1 \leq x_1 < \cdots < x_n \leq u$, we want to support the following fundamental operations:

* Funded by the Millennium Institute for Foundational Research on Data (IMFD).

- $\mathsf{rank}(S, x)$, which for $x \in U$, yields $|\{x_i \in S,\ x_i \leq x\}|$, and
- $\mathsf{select}(S, k)$, which for $k \in \mathbb{N}$, yields $x \in S$ such that $\mathsf{rank}(S, x) = k$.

If S is viewed as its characteristic bit vector (cbv for short) $C_S[1..u]$, such that $C_S[i] = \mathbf{1}$ iff $i \in S$, the problem becomes that of representing a bit vector with operations $\mathsf{rank}_{\mathbf{1}}(C_S, x)$, which yields the number of bits $\mathbf{1}$ in $C_S[1..x]$, and $\mathsf{select}_{\mathbf{1}}(C_S, k)$, that finds the position of the kth $\mathbf{1}$ bit in C_S. These are arguably the most fundamental building block of succinct data structures [28]. Also, they allow one to compute the fundamental operation $\mathsf{predecessor}(S, x) \equiv \mathsf{select}(S, \mathsf{rank}(S, x-1))$ (among others). We assume the transdichotomous word RAM model with word size $w = \mathrm{O}(\lg u) = \Omega(\lg n)$. Arithmetic, logic, and bitwise operations, as well as accesses to w-bit memory cells, take constant time.

Succinct data structures use space close to the corresponding information-theoretic lower bound while supporting operations efficiently. For instance, there are $\binom{u}{n}$ different subsets of U of size n. Hence, the information-theoretic lower bound on the number of bits needed to represent any such sets is $\mathcal{B}(n, u) = \left\lceil \lg \binom{u}{n} \right\rceil$ bits. If $n \ll u$, $\mathcal{B}(n, u) \approx n \lg e + n \lg \frac{u}{n} - \mathrm{O}(\lg u)$ bits. Compressed data structures, on the other hand, exploit regularities in specific instances of data to go below the information-theoretic lower bound. The space usage of compressed data structures on a specific instance is evaluated relative to some measure of compressibility of that instance.

Starting with the seminal work of Jacobson [25], much effort has gone into representing sets succinctly while supporting rank and select in $\mathrm{O}(1)$ time. Clark and Munro [9, 10] were the first to achieve $\mathrm{O}(1)$ time rank and select, using $u + \mathrm{O}(u/\lg\lg u)$ bits of space. Raman et al. [36] achieved succinct space, $\mathcal{B}(n, u) + \mathrm{O}(u \lg\lg u/\lg u)$ bits, with $\mathrm{O}(1)$-time operations. In order to capture the compressibility of sets better, researchers have considered the *empirical higher-order entropy* of C_S, denoted by $H_k(C_S)$, which achieves good compression (beyond $\mathcal{B}(n, u)$). Several researchers, including Sadakane and Grossi [37] showed how to achieve constant-time operations while using $uH_k(S) + \mathrm{O}(u((k+1) + \lg\lg u)/\lg u)$ bits. An important drawback is that for big universes (i.e., $n \ll u$), $H_k(C_S)$ decreases slowly as k grows. In addition, using small k it is not possible to capture longer-range dependencies (e.g. for any (long) string x, $H_k(x) \approx H_k(xx)$).

Another well-studied measure is $\mathtt{GAP}(S)$. If $S = \{x_1, \ldots, x_n\}$, with $x_1 < \ldots < x_n$, then define $g_1 = x_1 - 1$ and, for $i > 1$, $g_i = x_i - x_{i-1} - 1$, and

$$\mathtt{GAP}(S) = \sum_{i=1}^{n} \{\lfloor \lg g_i \rfloor + 1\}.$$

Although $\mathtt{GAP}(S)$ is not an *achievable* measure[4], $\mathtt{GAP}(S)$ exploits variation in the gaps between elements. It can be seen that $\mathtt{GAP}(S) < \mathcal{B}(n, u)$, and $\mathtt{GAP}(S)$ approaches $\mathcal{B}(n, u)$ only when $g_i = \frac{u}{n}$ (for $i = 1, \ldots, n$). Gupta et al. [23] showed how to represent S in $\mathtt{GAP}(S) + \mathrm{O}(n \lg \frac{u}{n}/\mathrm{poly}\lg(n)) \leq \mathcal{B}(n, u)(1 + o(1))$ bits while supporting rank and select quickly (albeit not in constant time).

[4] For example, if we choose every element in U to be in S with probability 0.5, then $\mathtt{GAP}(S) \sim 0.81u$, less than the Shannon lower bound for S.

In this paper we focus on applications where set elements are clustered, forming runs of successive elements. Some applications are interval intersection in computational biology [34], web-graph compression [3, 4], IR and query processing in reordered databases [26, 2], valid-time joins in temporal databases [39, 16, 13, 5], ancestor checking in trees [6, 7], data structures for set intersection [8], and bit vectors of wavelet trees [22, 14] of the Burrows-Wheeler transform of highly-repetitive texts [27, 24, 15]. Although $\texttt{GAP}(S)$ addresses this kind of non-uniform distribution, in the presence of runs, run-length encoding (RLE) [18] is more appropriate. Here, a set S with cbv $C_S[1..u] = \mathbf{0}^{z_1}\mathbf{1}^{l_1}\mathbf{0}^{z_2}\mathbf{1}^{l_2}\cdots\mathbf{0}^{z_g}\mathbf{1}^{l_g}$ is represented through the sequences $\langle z_1, \ldots, z_g\rangle$ and $\langle l_1, \ldots, l_g\rangle$ of $2g$ lengths of the alternating 0/1-runs in C_S (assume wlog that C_S begins with $\mathbf{0}$ and ends with $\mathbf{1}$). Then:

$$\texttt{RLE}(S) = \sum_{i=1}^{g} \{\lfloor \lg (z_i - 1)\rfloor + 1\} + \sum_{i=1}^{g} \{\lfloor \lg (l_i - 1)\rfloor + 1\}.$$

It holds that if $n < u/2$, $\texttt{RLE}(S) < \mathcal{B}(n, u) + n + \mathrm{O}(1)$ [14]. Note that $\texttt{RLE}(S)$ is also not an achievable measure, but handles sets S that contain runs better than $\texttt{GAP}(S)$—a set S with cbv $\mathbf{0}^{u-n}\mathbf{1}^n$ has $\texttt{GAP}(S) = \Theta(n + \lg u)$ but $\texttt{RLE}(S) = \Theta(\lg n + \lg u)$.

In practice, $\texttt{GAP}(S)$, $\texttt{RLE}(S)$ and H_k each perform well on specific data sets S. In the important case when S is a posting list in an inverted index, a recent breakthrough [30] showed that so-called *partitioned* Elias-Fano (PEF) indices are very effective in compressing sets and can support select in O(1) time. However, we are not aware of any compressibility measure associated with these indices, and it appears rank cannot be supported in constant time.

Since we wish not only to compress sets, but also to support operations on them, the *overall* space usage (including any space needed to support operations) is important. Firstly, since predecessor queries can be answered using rank and select, any lower bound for the former applies also to the joint use of the latter operations. Pătraşcu and Thorup [33] showed that if we use $\Theta(s \lg u)$ bits of space, the time to answer predecessor queries is given by:

$$\texttt{PT}(u, n, a) = \Theta(\min\{\tfrac{\lg n}{\lg\lg u}, \lg \tfrac{\lg (u/n)}{a}, \lg \tfrac{\lg u}{a} / \lg \tfrac{a \lg \frac{\lg u}{a}}{\lg n}, \lg \tfrac{\lg u}{a} / \lg \tfrac{\lg \frac{\lg u}{a}}{\lg \frac{\lg n}{a}}\}),$$

where $a = \lg \frac{s \lg u}{n}$. It follows from this that even if we are allowed to use $\mathrm{O}(\text{poly } n)$ words of space, constant-time operations are possible only for relatively small universes, i.e. $u = \mathrm{O}(n \cdot \text{poly}\lg(n))$, or for very small sets, i.e. $n = (\lg u)^{\mathrm{O}(1)}$. Fortunately, the first case, which is our main focus, is also very commonly seen in applications.

A more refined analysis looks at the *redundancy* of a data structure, which is the space used by a data structure over and above the corresponding space bound for representing the set itself. Pătraşcu [32] improved on earlier work [19, 21] and showed the following:

Theorem 1 ([32]). *For any $c > 0$, a set S can be represented using $\mathcal{B}(n, u) + \mathrm{O}(u/\lg^c (u/c)) + \mathrm{O}(u^{3/4}\text{poly}\lg(u))$ bits and support* rank *and* select *in $\mathrm{O}(c)$ time.*

We will use Theorem 1 only when $c = O(1)$. For the parameter values of interest, namely $u = O(n \cdot \text{poly} \lg(n))$ and $c = O(1)$, the redundancy of Theorem 1 was shown to be optimal by Pătraşcu and Viola [31]. In the so-called *systematic* model, [20] gave matching upper and lower bounds on redundancy.

Contributions. Our contributions are as follows. Firstly, in Sections 2 and 3, we give a surprisingly simple adaptive approach that stores S in potentially better than $\mathcal{B}(n, u)$ space, while still supporting constant-time rank/select. This is based on the intuition that within the class $\mathcal{C}_n$ of $\binom{u}{n}$ distinct sets of n elements from U, there are sets that can be represented more succinctly than others. For instance, in a extreme case where the elements form a single interval $[i..i + n - 1]$ of size n, why would one use $\lg \binom{u}{n}$ bits to describe this set? The smallest set element i and the set size n are enough to represent such a set.

Let class $\mathcal{C}_g^n \subset \mathcal{C}_n$ contain the sets whose elements are arranged in $g \leq n$ runs of $l_i \geq 1$ successive elements from U, for $i = 1, \ldots, g$. Also, let $\mathcal{C}_{g,r}^n \subset \mathcal{C}_g^n$ be a further refinement of class $\mathcal{C}_g^n$, which contains all sets that consist of g runs, such that $r \leq g$ of them have at least 2 elements.

- We introduce new compressibility measures for sets:
  - $\mathcal{L}_1 = \lg |\mathcal{C}_g^n| = \lg \binom{u-n+1}{g} + \lg \binom{n-1}{g-1}$ and
  - $\mathcal{L}_2 = \lg |\mathcal{C}_{g,r}^n| = \lg \binom{u-n+1}{g} + \lg \binom{n-g-1}{r-1} + \lg \binom{g}{r}$

  We show that $\mathcal{L}_2 \leq \mathcal{L}_1 \leq \mathcal{B}(n, u)$.
- We give data structures that use space close to bounds $\mathcal{L}_1$ and $\mathcal{L}_2$, namely $\lg \binom{u}{g} + \lg \binom{n}{g} + o(u) \approx \mathcal{L}_1 + o(u)$ bits of space and $\lg \binom{u}{g} + \lg \binom{n}{r} + \lg \binom{g}{r} + o(u) \approx \mathcal{L}_2 + o(u)$ bits of space, and support rank and select in $O(1)$ time.

Next, in Section 4, we revisit $\texttt{GAP}(S)$ and $\texttt{RLE}(S)$ measures in the following sense. The $\texttt{GAP}(S)$ and $\texttt{RLE}(S)$ measures encode a gap/run of length x using $1 + \lfloor \lg x \rfloor$ bits respectively. By Shannon's theorem, coding x using $1 + \lfloor \lg x \rfloor$ bits is tailoring the code length to a particular, and fixed, distribution of gap/run lengths[5]. We therefore propose two new measures of compressibility: we encode gap sizes and run lengths using their empirical zeroth-order entropy. That is, we treat the sequence of runs as a string of length $2g$ from the alphabet $[1..n]$, and encode each run using the Shannon optimal number of bits based upon the number of times this run length is seen (and analogously for gaps). On any set S, such approaches should outperform $\texttt{RLE}(S)$ and $\texttt{GAP}(S)$. For example, given a set S with $C_S = \mathbf{0}^4\mathbf{1}^{20}\mathbf{0}^4\mathbf{1}^{20} \cdots \mathbf{0}^4\mathbf{1}^{20}$, $\texttt{RLE}(S)$ is far inferior to H_0-coding the runs, which would use only one bit for encoding each run. We introduce two new measures of compressibility, $H_0^{\text{run}}(S)$ and $H_0^{\text{gap}}(S)$, to address this. We give data structures that support rank/select in O(1) time using space close to $H_0^{\text{gap}}(S)$, and select on both S and its complement in O(1) time using space close to $H_0^{\text{run}}(S)$. In this section, we also give additional compressibility measures.

Finally, in Section 5, we show experimentally that these approaches are promising for real-world datasets.

[5] Since $\texttt{GAP}(S)$ and $\texttt{RLE}(S)$ are not achievable, this statement is imprecise.

2 Adaptive Succinctness

We prove adaptive lower bounds for space needed to represent a set S.

Given a set $S = \{x_1, \ldots, x_n\} \subseteq U$ of n elements $1 \leq x_1 < \cdots < x_n \leq u$, a *maximal run of successive elements* $G \subseteq S$ contains $|G| \geq 1$ elements $x_i, x_i+1, \ldots, x_i+|G|-1$, such that $x_i - 1 \notin S$ and $x_i + |G| \notin S$. Let $G_1, \ldots, G_g$ be the partition of $S = \{x_1, \ldots, x_n\}$ into maximal runs of successive elements, such that $\forall x \in G_i, \forall y \in G_j$, $x < y \Leftrightarrow i < j$. Let us assume that $r \leq g$ of these G_i are of size $|G_i| \geq 2$.

Let $\mathcal{C}_n$ denote the class of sets of n elements from U. Notice that $|\mathcal{C}_n| = \binom{u}{n}$. We define class $\mathcal{C}_g^n \subset \mathcal{C}_n$, a refinement of $\mathcal{C}_n$ containing sets whose n elements are arranged in $g \leq n$ maximal runs of successive elements. Notice that $\bigcup_{g=1}^{n} \mathcal{C}_g^n = \mathcal{C}_n$. Let class $\mathcal{C}_{g,r}^n \subset \mathcal{C}_g^n$ be a further refinement consisting of all sets such that $r \leq g$ out of the g maximal runs have size ≥ 2. It holds that $\bigcup_{r=1}^{g} \mathcal{C}_{g,r}^n = \mathcal{C}_g^n$.

The cbv C_S of set $S \in \mathcal{C}_{g,r}^n$ consists of g 0-runs of lengths $z_1, \ldots, z_g$ such that $z_1 = \min\{G_1\} - 1$, and $z_i = \min\{G_i\} - \max\{G_{i-1}\} - 1$, and g 1-runs of length $l_i = |G_i|$, for $i = 1, \ldots, g$. We call *solitary* the elements in a run of size 1. Let $o_1 = l_{j_1} - 1, \ldots, o_r = l_{j_r} - 1$, for $j_1 < \cdots < j_r$, denote the sorted sequence of lengths (-1) of the r runs of length $l_{j_i} \geq 2$.

We show that within $\mathcal{C}_n$, there are sets that can be represented more succinctly than others, depending on which subclass they belong to. Less bit are needed to represent S if $S \in \mathcal{C}_g^n$ or, moreover, $S \in \mathcal{C}_{g,r}^n$. This is because the number of different such sets is smaller than $\binom{u}{n}$, so distinguishing them is easier. For instance, for $u = 6$ and $n = 3$ there are $\binom{6}{3} = 20$ different sets, yet if $g = 3$, the number of sets is only 4 ($\{1,3,5\}, \{1,3,6\}, \{1,4,6\}, \{2,4,6\}$). We formalize this fact next:

Theorem 2. *There are* $|\mathcal{C}_g^n| = \binom{n-1}{g-1}\binom{u-n+1}{g}$ *different sets whose* n *elements from* U *can be partitioned into* g *maximal runs of successive elements.*

Hence, we have:

Corollary 1. $\mathcal{L}_1 = \lg|\mathcal{C}_g^n| = \lg\binom{u-n+1}{g} + \lg\binom{n-1}{g-1}$ *bits are necessary to represent any set* $S \in \mathcal{C}_g^n$.

Next, we determine $|\mathcal{C}_{g,r}^n|$.

Theorem 3. *There are* $|\mathcal{C}_{g,r}^n| = \binom{n-g-1}{r-1}\binom{g}{r}\binom{u-n+1}{g}$ *different sets whose* n *elements from* U *can be partitioned into* g *maximal runs of successive elements, such that* $r \leq g$ *of these groups have at least 2 elements.*

Corollary 2. $\mathcal{L}_2 = \lg|\mathcal{C}_{g,r}^n| = \lg\binom{u-n+1}{g} + \lg\binom{n-g-1}{r-1} + \lg\binom{g}{r}$ *bits are necessary to represent any set* $S \in \mathcal{C}_{g,r}^n$.

3 Adaptive Succinct rank/select Data Structures

Given set $S \in \mathcal{C}_{g,r}^n$, let us define its *essential* sets:

1. $\hat{P} = \{p_1, \ldots, p_g\} \subseteq S$ such that $p_i = \min\{G_i\}$, for $i = 1, \ldots, g$. This set has universe u. We call each element p_i a *pioneer*;
2. $\hat{L} = \{l_1, \ldots, l_g\}$, such that $l_j = \sum_{i=1}^{j} |G_i|$. The corresponding characteristic bit vector is $C_{\hat{L}} = \mathbf{0}^{|G_1|-1}\mathbf{1}\mathbf{0}^{|G_2|-1}\mathbf{1}\cdots\mathbf{0}^{|G_g|-1}\mathbf{1}$ of length (universe of $\hat{L}$) n and g **1**s. These are the unary encodings of $|G_i|$s.
3. $\hat{G} = \{y_1, \ldots, y_g\}$ such that $y_j = \sum_{i=1}^{j} z_i$, for $j = 1, \ldots, g$. The corresponding characteristic bit vector is $C_{\hat{G}} = \mathbf{0}^{z_1-1}\mathbf{1}\mathbf{0}^{z_2-1}\mathbf{1}\cdots\mathbf{0}^{z_g-1}\mathbf{1}$, of length (universe of $\hat{G}$) $\sum_{i=1}^{g} z_i = u - n$ (i.e., the number of **0**s in C_S), and it has g **1**s. These are the unary encodings of values z_i.
4. $\hat{R} = \{q_1, \ldots, q_r\}$ such that $q_j = \sum_{i=1}^{j} o_i$, for $j = 1, \ldots, r$. The corresponding characteristic bit vector is $C_{\hat{R}} = \mathbf{0}^{o_1-1}\mathbf{1}\mathbf{0}^{o_2-1}\mathbf{1}\cdots\mathbf{0}^{o_r-1}\mathbf{1}$, of length (universe of $\hat{R}$) $n - g$, and r **1**s. These are the unary encodings of values o_i.
5. $\hat{V} = \{v_1, \ldots, v_r\}$ such that $|G_{v_i}| > 1$, for $i = 1, \ldots, r$. The corresponding characteristic bit vector $C_{\hat{V}}[1..g]$ has length g and r **1**s. It holds that $C_{\hat{V}}[i] =$ **1** iff the run G_i of the ith pioneer has $|G_i| > 1$.

A set S can be unambiguously described with the following combinations of essential sets: (Scheme 1) Sets $\hat{P}$ and $\hat{L}$; (Scheme 2) Sets $\hat{P}$, $\hat{R}$, and $\hat{V}$; (Scheme 3) Sets $\hat{G}$, $\hat{R}$, and $\hat{V}$; and (Scheme 4) Sets $\hat{G}$ and $\hat{L}$. Notice that, for instance, the Elias-Fano encoding of sets $\hat{G}$ and $\hat{L}$ (Scheme 4) yields space close to $\mathcal{L}_1$. The Elias-Fano encoding of Scheme 3, alternatively, yields space close to $\mathcal{L}_2$. In what follows, we build on above schemes to obtain adaptive succinct data structures.

3.1 Using Space Close to $\mathcal{L}_1$

Given a set $S \in \mathcal{C}_g^n$, consider the interval $[p_i..p_{i+1})$, for $1 \leq i < g$, between two consecutive pioneers. This is the *locus* of pioneer p_i [12]: all $\mathsf{rank}(S, x)$ queries within this interval (i.e., $p_i \leq x < p_{i+1}$) have similar answer, obtainable from p_i and $|G_i|$ (similarly for select). We use Scheme 1 above, which builds on sets $\hat{P}$ and $\hat{L}$. Building on Scheme 4 would use space closer to $\mathcal{L}_1$, however it does not allow (seemingly) for constant-time $\mathsf{rank}/\mathsf{select}$.

Operation $\mathsf{rank}(S, x)$. Notice that $\forall x$ such that $p_i \leq x < p_i + |G_i|$, $\mathsf{rank}(S, x) \equiv \mathsf{rank}(S, p_i) + x - p_i$; otherwise, if $p_i + |G_i| \leq x < p_{i+1}$, $\mathsf{rank}(S, x) \equiv \mathsf{rank}(S, p_i) + |G_i| - 1$. So, we show how to compute p_i, $\mathsf{rank}(S, p_i)$, and $|G_i|$ from sets $\hat{P}$ and $\hat{L}$. Let $i = \mathsf{rank}(\hat{P}, x)$ be the number of pioneers that are smaller (or equal) than x, then $p_i = \mathsf{select}(\hat{P}, i)$. Hence, $\mathsf{rank}(S, p_i) \equiv \mathsf{select}(\hat{L}, i-1) + 1$, since $\mathsf{select}(\hat{L}, i-1) = \sum_{j=1}^{i-1} |G_j|$. Finally, $|G_i| = \mathsf{select}(\hat{L}, i) - \mathsf{select}(\hat{L}, i-1)$.

Operation $\mathsf{select}(S, k)$. Assume that for the element x_k we are looking for, it holds that $p_i \leq x_k < p_{i+1}$. Then, $\mathsf{select}(S, k) \equiv p_i + k - \mathsf{rank}(S, p_i)$. This time, $i = \mathsf{rank}(\hat{L}, k) + [k \notin \hat{L}]$ [6], and $p_i = \mathsf{select}(\hat{p}, i)$. Finally, as explained above for operation rank, $\mathsf{rank}(S, p_i) \equiv \mathsf{select}(\hat{L}, i-1) + 1$.

We represent $\hat{P}$ and $\hat{L}$ using Theorem 1. This uses $\lg\binom{u}{g} + \lg\binom{n}{g} + \mathrm{O}(u/\lg^c u)$ bits, for any constant $c \geq 1$, and supports rank and select in $\mathrm{O}(1)$ time.

[6] $[k \notin \hat{L}]$ is Iverson brackets notation, which equals 1 iff $k \notin \hat{L}$ is true, 0 otherwise.

Theorem 4. *There exists a data structure that represents any set $S \in \mathcal{C}_g^n$ of n elements from universe U, using $\lg \binom{u}{g} + \lg \binom{n}{g} + \mathrm{O}(u/\lg^c u)$ bits, for any constant $c \geq 1$, while supporting operations* rank *and* select *in* $\mathrm{O}(1)$ *time.*

3.2 Using Space Close to $\mathcal{L}_2$

We use sets $\hat{P}$, $\hat{V}$, and the following variant of set $\hat{R}$: $\hat{R}' = \{q'_1, \ldots, q'_r\}$ such that $q'_j = \sum_{i=1}^{j} (o_i + \mathsf{select}(\hat{V}, i) - \mathsf{select}(\hat{V}, i-1))$. This set has universe $[1..n]$, and has r elements. To understand how this set works, let us see at its characteristic bit vector $C_{\hat{R}'}[1..n]$. It has r **1**s, each corresponding to a run G_i. Each such **1** is preceded by $|G_i| - 1$ **0**s. Consider runs G_i and G_{i+l}, for $i, l \geq 1$, such that runs $G_{i+1}, \ldots, G_{i+l-1}$ are each of size 1 (i.e., they correspond to solitary elements in S). Then, in $C_{\hat{R}'}$ there are $l - 1 + |G_i| - 1$ **0**s between the **1**s corresponding to G_i and G_{i+l}.

Operation $\mathsf{rank}(S, x)$. Let us assume that $p_i \leq x < p_{i+1}$, for $p_i \in \hat{P}$. First, consider the case where $|G_i| = 1$. That is, p_i is a solitary pioneer. Notice that $\mathsf{rank}(S, x) \equiv \mathsf{rank}(S, p_{i-l}) + |G_{i-l}| - 1 + l$, for $l \geq 1$, such that p_{i-l} is the greatest pioneer smaller than p_i such that $|G_{i-l}| > 1$ (assume $p_{i-l} = 0$ if there is none). Let $i = \mathsf{rank}(\hat{P}, x)$, and $p_i = \mathsf{select}(\hat{P}, i)$. Then, $l = i - \mathsf{select}(\hat{V}, \mathsf{rank}(\hat{V}, i))$. Hence, $\mathsf{rank}(S, p_{i-l}) + |G_{i-l}| - 1 \equiv \mathsf{select}(\hat{R}', \mathsf{rank}(\hat{V}, i))$, and we are done. Otherwise, $|G_i| > 1$, so we must distinguish two cases: (1) $p_i \leq x < p_i + |G_i|$, in whose case $\mathsf{rank}(S, x) \equiv \mathsf{rank}(S, p_i) + x - p_i$; or (2) $p_i + |G_i| \leq x$, hence $\mathsf{rank}(S, x) \equiv \mathsf{rank}(S, p_i) + |G_i| - 1$.Notice that $\mathsf{rank}(S, p_i) \equiv \mathsf{rank}(S, p_{i-l}) + |G_{i-l}| - 1 + l$, which has been already computed. Finally, $|G_i| \equiv \mathsf{select}(\hat{R}', \mathsf{rank}(\hat{V}, i)) - \mathsf{select}(\hat{R}', \mathsf{rank}(\hat{V}, i) - 1) - l$.

Operation $\mathsf{select}(S, k)$. We must determine whether x_k is a solitary element or lies within a run of successive elements of length > 1. Let us regard runs G_v and G_i, both of size > 1, such that there is no other run of size > 1 between them, and $p_v + |G_v| - 1 < x_k \leq p_i + |G_i| - 1$. Here, $j = \mathsf{rank}(\hat{R}', k) - [k \in \hat{R}']$ and $v = \mathsf{select}(\hat{V}, j)$. The number of solitary pioneers between runs G_v and G_i is $l = \mathsf{select}(\hat{V}, j+1) - \mathsf{select}(\hat{V}, j) - 1$. Let $s = \mathsf{select}(\hat{R}', j)$ be the rank up to position $p_v + |G_v| - 1$ (i.e., up to the last element in G_v). Notice that if $k - s \leq l$, x_k is the $(k-s)$th pioneer after G_v Otherwise, if $k - s > l$, x_k lies within G_i.

We represent sets $\hat{P}$, $\hat{V}$, and $\hat{R}'$ using Theorem 1 and obtain:

Theorem 5. *There exists a data structure that represents any set $S \in \mathcal{C}_{g,r}^n$ of n elements from universe U, using $\lg \binom{u}{g} + \lg \binom{n}{r} + \lg \binom{g}{r} + \mathrm{O}(u/\lg^c u)$ bits, for any constant $c \geq 1$, while supporting operations* rank *and* select *in* $\mathrm{O}(1)$ *time.*

4 Further Squeezing rank/select Data Structures

In this section, we study the extent to which a static rank/select data structure can be squeezed, while still supporting operations efficiently. Let t_r denote the

time complexity of operation rank, and t_s that of select. Since predecessor(S, x) can be reduced to select$(S, \mathsf{rank}(S, x-1))$, Pătraşcu and Thorup's [33] predecessor lower bound is also a lower bound for $t_r + t_s$. It is natural, then, to compare with this lower bound to see how the time deteriorates as we squeeze.

4.1 GAP(S)

Gupta et al. [23] introduce a data structure using $\mathtt{GAP}(S) + \mathrm{O}(n \lg \frac{u}{n} / \lg n) + \mathrm{O}(n \lg\lg \frac{u}{n})$ bits of space. Originally, Andersson and Thorup's predecessor data structure [1] is used as building block (using $\mathrm{O}(n \lg \frac{u}{n} / \lg n)$ bits of space), yet it can be easily modified to use Pătraşcu and Thorup's data structure [33]; to get the same space usage, we set $a = \lg(\lg u / \lg^2 n)$. Operation select is supported in $\mathrm{O}(\lg\lg n)$ time, and rank in $\mathtt{PT}(u, n, \lg(\lg u / \lg^2 n)) + \mathrm{O}(\lg\lg n)$ time.

4.2 RLE(S)

We now consider $\mathtt{RLE}(S)$, and begin by noting that $\mathtt{RLE}(S) = \mathtt{GAP}(\hat{G}) + \mathtt{GAP}(\hat{L})$.

Property 1. For any set $S \in \mathcal{C}_g^n$ it holds that $\mathtt{RLE}(S) \leq \lg \binom{u-n+1}{g} + \lg \binom{n}{g}$.

Since $\mathtt{RLE}(S) = \mathtt{GAP}(\hat{G}) + \mathtt{GAP}(\hat{L})$, the proof is immediate. Set $\hat{G}$ has g elements over universe of size $u - n$, it holds that $\mathtt{GAP}(\hat{G}) \leq \lg \binom{u-n}{g} \leq \lg \binom{u-n+1}{g}$. Similarly, $\mathtt{GAP}(\hat{L}) \leq \lg \binom{n}{g}$.

Theorem 6. *There exists a data structure that represents any set $S \in \mathcal{C}_g^n$ over the universe U, using $\mathtt{RLE}(S) + \mathrm{O}(g \lg \frac{u}{g} / \lg g) + \mathrm{O}(g \lg\lg \frac{u}{g})$ bits of space, and supporting operation* select *in $\mathtt{PT}(u-n, g, \lg \frac{\lg u - n}{\lg^2 g})$ time, whereas operation* rank *is supported in $\mathtt{PT}(u-n, g, \lg \frac{\lg u - n}{\lg^2 g}) + \mathrm{O}((\lg\lg g)^2)$ time.*

This is achieved by using the data structure from Section 4.1 on sets $\hat{G}$ and $\hat{L}$.

4.3 H_0 coding of gaps and runs

In this section, we first describe our new measures of compressing sets based on H_0 coding the gaps or runs (called $H_0^{\mathrm{gap}}(S)$ and $H_0^{\mathrm{run}}(S)$). We then describe the main result of this section: a data structure that supports rank and select on a set S in constant time while using close to $H_0^{\mathrm{gap}}(S)$ space. We then obtain as a corollary a representation of S using close to $H_0^{\mathrm{run}}(S)$ space, but supporting only select on S and $U \setminus S$. Finally, we relate these measures to Theorem 3.

Definition 1. *Let $S = \{x_1, \ldots, x_n\}$, with $1 < x_1 < \ldots < x_n = u$. Define $g_1 = x_1 - 1$ and, for $i > 1$, $g_i = x_i - x_{i-1}$. Let $\mathcal{G}(S) = \{h_1^{m(h_1)}, h_2^{m(h_2)}, \ldots, h_t^{m(h_t)}\}$ be the multiset of values in the sequence $\langle g_1, \ldots, g_n \rangle$, where $m(h_i)$ is the multiplicity of h_i in the sequence of gaps. Then:*

$$nH_0^{\mathrm{gap}}(S) = \lg \binom{n}{m(h_1), m(h_2), \ldots, m(h_t)}.$$

Letting $\hat{L}$ and $\hat{G}$ be as defined at the start of Section 3, we define:

$$H_0^{\mathrm{run}}(S) = H_0^{\mathrm{gap}}(\hat{L}) + H_0^{\mathrm{gap}}(\hat{G}).$$

Remark. Note that $H_0^{\text{gap}}(S)$ is (almost) an achievable measure: we can apply arithmetic coding to the sequence of gaps, which would take $H_0^{\text{gap}}(S)$ bits. However, we would also need to output the model of the arithmetic coder, which can be $\mathcal{G}(S)$ itself, stored in $t(\lg u + \lg n)$ bits, specifying which runs are present and their multiplicities. Since $t = \mathrm{O}(\sqrt{u})$, this is not excessive for many applications. Similar remarks apply to $H_0^{\text{run}}(S)$. Observe that $|\hat{L}| = |\hat{G}| = g$, so we would aim to compress S to $gH_0^{\text{run}}(S)$ bits.

$H_0^{\text{gap}}(S)$. In this section we show the following:

Theorem 7. *Given a set $S \subseteq [1..u]$, $|S| = n$, we can represent it to support* rank *and* select *in* $\mathrm{O}(1)$ *time using $nH_0^{\text{gap}}(S) + \mathrm{O}(n) + o(u)$ bits.*

Proof. Let the elements of S be $\{x_i\}_{i=1}^n$, sequence of gaps be $\{g_i\}_{i=1}^n$, and the multiset of gaps be $\mathcal{G}(S) = \{h_i^{m(h_i)}\}_{i=1}^t$. We first note that the result is achieved trivially if $n = \mathrm{O}(u/\lg u)$: we represent S as the bit-string $\mathbf{0}^{g_1-1}\mathbf{1}\ldots\mathbf{0}^{g_n-1}\mathbf{1}$, which is of length u and has n $\mathbf{1}$s. If stored using Theorem 1, this bit-string will use $\frac{u}{B}\lg\frac{u}{nB} + \mathrm{O}(\frac{u}{B}) + \frac{u}{(\lg u)^{\mathrm{O}(1)}} = \mathrm{O}(u\lg\lg u/\lg u) = o(u)$ bits and rank and select operations can be supported in $\mathrm{O}(1)$ time, which proves the theorem. We therefore henceforth assume that $n \geq cu/\lg u$ for some sufficiently large $c \geq 1$.

We begin by converting S to a new set S' with $n' \leq n + u/B = \mathrm{O}(n)$ numbers from U, for some integer parameter $B = \Theta((\lg n/\lg\lg n)^2)$, by setting $S' = S \cup \{iB | 1 \leq i \leq u/B\}$. Let the elements, the sequence of gaps, and the multiset of gaps of S' be $\{x'_i\}_{i=1}^{n'}$, $\{g'_i\}_{i=1}^{n'}$, and $\mathcal{G}(S') = \{h_i'^{m'(h'_i)}\}_{i=1}^{t'}$. We now show that $H_0^{\text{gap}}(S')$ is close to $H_0^{\text{gap}}(S)$, using the following [11, Theorem 17.3.3]:

Theorem 8. *Let p and q be two probability mass functions on a set T such that $||p-q||_1 = \sum_{x\in T}|p(x)-q(x)| \leq \frac{1}{2}$. Then $|H(p) - H(q)| \leq ||p-q||_1 \lg \frac{|T|}{||p-q||_1}$.*

Let T be the underlying set of the multiset $\mathcal{G}(S) \cup \mathcal{G}(S')$. For any integer $x \in T$ let $p(x) = m(x)/n$ and $q(x) = m'(x)/n'$; $m(x) = 0$ if $x \notin \mathcal{G}(S)$, and and similarly $m'(x)$. It is easy to see that $||p-q||_1 = \mathrm{O}(u/(nB))$, since at most u/B gaps in S are changed during the conversion, and $n = \Theta(n')$. Since we assume $n \geq cu/\lg u$ for any constant c, we can ensure that $||p-q||_1 \leq 1/2$. Noting that $||p-q||_1 = \Omega(1/n)$ (unless $p = q$, in which case the RHS of Theorem 8 equals 0 as well), we see that $\lg\frac{|T|}{||p-q||_1} = \mathrm{O}(\lg n|T|) = \mathrm{O}(\lg u)$. It follows that $|H_0^{\text{gap}}(S) - H_0^{\text{gap}}(S')| = \mathrm{O}(\frac{u\lg u}{nB})$ and hence that $|nH_0^{\text{gap}}(S) - n'H_0^{\text{gap}}(S')| = \mathrm{O}(\frac{u\lg u}{B}) = \mathrm{O}(u\frac{(\lg\lg u)^2}{\lg u}) = o(u)$.

The data structure comprises two parts: first, we divide the bit-string representing S into blocks of B consecutive bits (the i-th block corresponds to the interval $[(i-1)B + 1..iB]$). Next, we create a bit-string O which encodes, for each block, the count of the number of elements of S that lie in each block, written in unary. O will have u/B $\mathbf{0}$s and n $\mathbf{1}$s. If stored using Theorem 1, O will use $\frac{u}{B}\lg\frac{u}{nB} + \mathrm{O}(\frac{u}{B}) + \frac{u}{(\lg u)^{\mathrm{O}(1)}} = \mathrm{O}(u\lg\lg u/\lg u) = o(u)$ bits and support rank and select operations on both $\mathbf{0}$ and $\mathbf{1}$ in $\mathrm{O}(1)$ time. It is easy to see (details

omitted) how to reduce rank/select operations on S to rank/select operations on individual blocks using O.

We now describe the representation of an individual block. Each block is encoded independently; by Jensen's inequality, if the gaps in each block are encoded using H_0 bits, the total space usage of all block encodings is $H_0^{\text{gap}}(S')$. We also need to store the arithmetic coding model for each block, which requires $\mathrm{O}(\sigma \lg B)$ bits, where $\sigma = \mathrm{O}(\sqrt{B})$ is the number of distinct integers in a block. The overhead of the models in each block is therefore $\mathrm{O}(\frac{u\sqrt{B}\lg B}{B}) = \mathrm{O}(u\frac{(\lg\lg u)^2}{\lg u})$ bits. We now fix $B = (c \lg u/\lg\lg u)^2$ for sufficiently small constant c, and make the following observations that allow all operations in a block to be performed in $\mathrm{O}(1)$ time using table lookup (details omitted): (i) all integers in a block are $\mathrm{O}(\lg\lg u)$ bits long (ii) if we group the integers in a block with into sub-blocks of $c \lg u/\lg\lg u$ bits, we can ensure that the H_0 code of a sub-block is no more than $c' \lg u$ bits long for some $c' < 1$ (iii) the encoding of the arithmetic coding model for each block also fits into $c' \lg n$ bits (this is important since to decode the encoding of a sub-block, we must also use the arithmetic coding model as an argument to the table lookup). □

$H_0^{\text{run}}(S)$. Given a set $S \subseteq U$, let $\hat{L}$ and $\hat{G}$ be defined as in Section 3. Applying Theorem 7 to $\hat{L}$ and $\hat{G}$ and using [35, Theorem 1(c)], we obtain:

Corollary 3. *Given a set $S \subseteq U = [1..u]$ such that $|S| = n$ and $|\hat{L}| = g$, S can be represented in $gH_0^{\text{run}}(S) + \mathrm{O}(g) + o(u)$ bits and support* select *on S and on $U \setminus S$ in $\mathrm{O}(1)$ time.*

Discussion. Theorem 7 and Corollary 3 refine the results from Section 4.1 and Theorem 6 in terms of the space bound. In Section 3.2, we described Scheme 4, which comprises the sets $\hat{P}$, $\hat{V}$, and $\hat{R}'$. An alternative view of Scheme 1 from Section 3.1, and how it leads to Scheme 4 and then towards $H_0^{\text{run}}(S)$ is as follows. Scheme 1 identifies the start positions of the runs of **1**s using $\hat{P}$, then encodes their lengths using $\hat{L}$; each run is encoded using $\lg(n/g) + O(1)$ bits, i.e., each run of **1**s is encoded using a number of bits equal to the log of the average run length. This is clearly non-optimal if the distribution of the lengths of the runs is non-uniform, which can happen in many situations (for example, in a random bit-string, run-lengths are geometrically distributed). Scheme 4 improves upon Scheme 1 by encoding runs of length 1 using the Shannon optimal number of bits, based upon the number of times run length 1 length is seen. Choosing to focus on runs of length 1 can be non-optimal: e.g. in a set where the runs were of length $1, 2, 2, \ldots, 2$, Scheme 4 would offer no improvement over Scheme 1. The next step, that we consider experimentally, is to modify Scheme 4 to encode $\hat{L}$ adaptively using Theorem 7. Such a modification should give superior compression to Scheme 4, while supporting $\mathrm{O}(1)$-time *rank*/*select*. Finally, $H_0^{\text{run}}(S)$ is the logical conclusion, replacing the "non-adaptive" encoding of $\hat{P}$ with an adaptive encoding of $\hat{G}$.

4.4 $\texttt{HYB}(S)$

Next, we study the following compression measure.

Definition 2. *Given set $S \in \mathcal{C}^n_{g,r}$, we define the entropy measure:*

$$\begin{aligned}\texttt{HYB}(S) &= \sum_{i=1}^{g} \{\lfloor \lg (z_i - 1) \rfloor + 1\} + \sum_{i=1}^{r} \{\lfloor \lg (o_i - 1) \rfloor + 1\} \\ &= \texttt{GAP}(\hat{G}) + \texttt{GAP}(\hat{R}).\end{aligned}$$

Similar to Property 1, we have:

Property 2. For any set $S \in \mathcal{C}^n_{g,r}$ it holds that $\texttt{HYB}(S) \leq \lg \binom{u-n+1}{g} + \lg \binom{n-g}{r}$.

Property 3. Given set $S \in \mathcal{C}^n_{g,r}$, $\texttt{HYB}(S) \leq \min\{\texttt{GAP}(S), \texttt{RLE}(S)\}$.

Proof omitted for lack of space.

Theorem 9. *There exists a data structure that represents any set $S \in \mathcal{C}^n_{g,r}$ over the universe U, using $\texttt{HYB}(S)(1 + o(1)) + \mathrm{O}(g \lg \frac{u-n}{g} / \lg g) + \mathrm{O}(r \lg \frac{n-g}{r} / \lg r) + \lg \binom{g}{r}$ bits. Operations* rank *and* select *are supported in $\texttt{PT}(u-n, \frac{g}{\lg^2 g}, \lg \frac{\lg u - n}{\lg^2 g}) + \mathrm{O}((\lg \lg g)^2)$ worst-case time.*

4.5 $\widetilde{\texttt{HYB}}(S)$

Next, we use gap compression on sets $\hat{P}$ and $\hat{L}$ to obtain space smaller than $\texttt{GAP}(S)$ in some cases, with query time that equals that of Section 4.1 (and hence improving Theorems 6 and 9).

Definition 3. *Given a set $S \in \mathcal{C}^n_{g,r}$ with pioneers $\hat{P} = \{p_1, \ldots, p_g\} \subseteq S$ and 1-run lengths $\hat{L} = \{l_1, \ldots, l_g\}$, we define the following compression measure:*

$$\begin{aligned}\widetilde{\texttt{HYB}}(S) &= \sum_{i=1}^{g} \{\lfloor \lg (p_i - p_{i-1} - 1) \rfloor + 1\} + \sum_{i=1}^{r} \{\lfloor \lg (l_i - 1) \rfloor + 1\} \\ &= \texttt{GAP}(\hat{P}) + \texttt{GAP}(\hat{L}).\end{aligned}$$

We can prove:

Lemma 1. *Given a set $S \in \mathcal{C}^n_{g,r}$, it holds that $\widetilde{\texttt{HYB}}(S) \leq \lg \binom{u}{g} + \lg \binom{n}{g}$.*

Theorem 10. *There exists a data structure that represents any set $S \in \mathcal{C}^n_{g,r}$ over the universe U, using $\widetilde{\texttt{HYB}}(S)(1+o(1)) + o(g \lg \frac{u}{g})$ bits of space, and supports operation* select *in $\mathrm{O}(\lg \lg g)$ time, and* rank *in $\texttt{PT}(u, g, \lg \frac{\lg u}{\lg^2 g}) + \mathrm{O}(\lg \lg g)$ time.*

Proof. Use the data structure from Section 4.1 to represent sets $\hat{P}$ and $\hat{L}$, and use the O(1)-time support for rank and select described in Section 3.1. □

5 Experimental Results

We show preliminary experiments on the space usage of the compressed approaches proposed in this paper. We use the URL-sorted GOV2 inverted index [38] as input, as this document order tends to generate runs in the posting lists. The universe size is 25,138,630 (this is the number of documents in the collection). We consider posting lists of size at least 100,000. We average the total space of each approach over 5,055,078,461 total postings. Table 1 shows the average number of bits per element for different state-of-the-art rank/select data structures (upper table) and different compression measures (bottom).

Table 1. Average number of bits per element for different rank/select data structures (on top) and compression measures (bottom), for URL-sorted GOV2 posting lists.

`sd_vector`	`rrr<127>`	`rrr<63>`	`rrr<31>`	`rrr<15>`	`PEF`	$\mathcal{L}_1$ d.s.	$\mathcal{L}_2$ d.s.
7.29	4.53	6.28	9.4	14.78	2.8	4.62	4.14

$\mathcal{L}_1$	$\mathcal{L}_2$	$\texttt{GAP}(S)$	$\texttt{RLE}(S)$	$\texttt{HYB}(S)$	$\widetilde{\texttt{HYB}}(S)$	$nH_0^{\text{gap}}(S)$	$gH_0^{\text{run}}(S)$
3.65	3.43	3.14	2.77	2.81	3.29	2.77	2.46

We considered the most efficient data structures from the `sdsl` library [17]: Elias-Fano `sd_vector` [29], Raman et al. [36] `rrr` data structure (using blocks of size 15, 31, 63, and 127). We also compared with the (very space-efficient) partitioned Elias-Fano approach [30] (`PEF` in the table). We used `sd_vector` to represent the sets that comprise the $\mathcal{L}_1$ and $\mathcal{L}_2$ data structures. The space for these data structures is reported at the top of Table 1.

In the bottom of Table 1, we report on compression measures applied to the above dataset (without making any allowance for space needed to support rank or select). Note that, as defined, the compression measures $\texttt{GAP}(S)$, $\texttt{RLE}(S)$, $\texttt{HYB}(S)$, and $\widetilde{\texttt{HYB}}(S)$ are not realizable. To make a fair comparison with a realizable compression measure, we assume that gaps/runs are encoded using Elias-δ coding, i.e. our reported $\texttt{GAP}(S)$ equals $\texttt{GAP}(S) = \sum_{i=1}^{n} \lfloor \lg g_i \rfloor + 2\lfloor \lg(\lfloor \lg g_i \rfloor + 1) \rfloor + 1$, and similarly for $\texttt{RLE}(S)$. Also, $\texttt{HYB}(S)$ includes space for the Elias-Fano representation of $\hat{V}$ (which uses space slightly more than $\lg \binom{g}{r}$ bits). As it can be seen, the results are promising in practice.

6 Conclusion and Open Problems

We have presented new measures of the compressibility of sets that are suitable when the elements of the sets are clustered in runs. In addition, when the sets are relatively dense (i.e. $n = u/(\lg u)^{\mathrm{O}(1)}$) we present data structures whose space usage is close to these measures, but which support rank and select operations in $O(1)$ time. Our preliminary experimental results show that our approaches yield space-efficient set representations.

There are a number of open directions that could be pursued. For example, we believe that an analogue of Theorem 7 for $\texttt{RLE}(S)$ is well within reach. Other interesting directions would be to close the gap between the space bounds $\mathcal{L}_1$ and $\mathcal{L}_2$ and their corresponding data structures. Finally, the data structure of Theorem 7 is unlikely to be practical; finding a practical variant with small redundancy is another interesting question.

References

1. A. Andersson and M. Thorup. Dynamic ordered sets with exponential search trees. *Journal of the ACM*, 54(3):13, 2007.
2. D. Arroyuelo, M. Oyarzún, S. González, and V. Sepulveda. Hybrid compression of inverted lists for reordered document collections. *Information Processing & Management*, 54(6):1308–1324, 2018.
3. P. Boldi and S. Vigna. The webgraph framework I: compression techniques. In *Proc. 13th international conference on World Wide Web (WWW)*, pages 595–602, 2004.
4. P. Boldi and S. Vigna. The webgraph framework II: codes for the world-wide web. In *Proc. Data Compression Conference (DCC)*, page 528, 2004.
5. F. Cafagna and M. H. Böhlen. Disjoint interval partitioning. *VLDB Journal*, 26(3):447–466, 2017.
6. Y. Chen and Y. Chen. An efficient algorithm for answering graph reachability queries. In *Proc. 24th International Conference on Data Engineering (ICDE)*, pages 893–902, 2008.
7. Y. Chen and Y. Chen. Decomposing dags into spanning trees: A new way to compress transitive closures. In *Proc. 27th International Conference on Data Engineering (ICDE)*, pages 1007–1018, 2011.
8. Y. Chen and W. Shen. An efficient method to evaluate intersections on big data sets. *Theoretical Computer Science*, 647:1–21, 2016.
9. D. Clark. *Compact PAT trees.* PhD thesis, University of Waterloo, 1997.
10. D. R. Clark and J. I. Munro. Efficient suffix trees on secondary storage (extended abstract). In *Proc. of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 383–391, 1996.
11. T. M. Cover and J. A. Thomas. *Elements of Infor.* Wiley Interscience, 2006.
12. M. de Berg, O. Cheong, M. J. van Kreveld, and M. H. Overmars. *Computational geometry: algorithms and applications, 3rd Edition.* Springer, 2008.
13. A. Dignös, M. H. Böhlen, and J. Gamper. Overlap interval partition join. In *Proc. Int. Conference on Management of Data (SIGMOD)*, pages 1459–1470, 2014.
14. L. Foschini, R. Grossi, A. Gupta, and J. S. Vitter. When indexing equals compression: Experiments with compressing suffix arrays and applications. *ACM Transactions on Algorithms*, 2(4):611–639, 2006.
15. T. Gagie, G. Navarro, and N. Prezza. Optimal-time text indexing in bwt-runs bounded space. In *Proc. 29h Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1459–1477, 2018.
16. D. Gao, C. S. Jensen, R. T. Snodgrass, and M. D. Soo. Join operations in temporal databases. *VLDB Journal*, 14(1):2–29, 2005.
17. S. Gog and M. Petri. Optimized succinct data structures for massive data. *Software: Practrice and Experience*, 44(11):1287–1314, 2014.

18. S. Golomb. Run-length encodings (corresp.). *IEEE Transactions on Information Theory*, 12(3):399–401, 1966.
19. A. Golynski, R. Grossi, A. Gupta, R. Raman, and S. S. Rao. On the size of succinct indices. In *Proc. 15th Annual European Symposium on Algorithms (ESA)*, volume 4698 of *LNCS*, pages 371–382. Springer, 2007.
20. Alexander Golynski, Alessio Orlandi, Rajeev Raman, and S. Srinivasa Rao. Optimal indexes for sparse bit vectors. *Algorithmica*, 69(4):906–924, 2014.
21. Alexander Golynski, Rajeev Raman, and S. Srinivasa Rao. On the redundancy of succinct data structures. In Joachim Gudmundsson, editor, *Algorithm Theory - SWAT 2008, 11th Scandinavian Workshop on Algorithm Theory, Gothenburg, Sweden, July 2-4, 2008, Proceedings*, volume 5124 of *Lecture Notes in Computer Science*, pages 148–159. Springer, 2008.
22. R. Grossi, A. Gupta, and J. S. Vitter. High-order entropy-compressed text indexes. In *Proc. 14th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 841–850, 2003.
23. A. Gupta, W.-K. Hon, R. Shah, and J. S. Vitter. Compressed data structures: Dictionaries and data-aware measures. *Theoretical Computer Science*, 387(3):313–331, 2007.
24. H. Huo, L. Chen, H. Zhao, J. S. Vitter, Y. Nekrich, and Q. Yu. A data-aware fm-index. In *Proc. 17th Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 10–23, 2015.
25. G. Jacobson. Space-efficient static trees and graphs. In *Proc. 30th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 549–554, 1989.
26. D. S. Johnson, S.r Krishnan, J. Chhugani, S. Kumar, and S. Venkatasubramanian. Compressing large boolean matrices using reordering techniques. In *Proc. 30th I(nternational Conference on Very Large Data Bases (VLDB)*, pages 13–23, 2004.
27. V. Mäkinen and G. Navarro. Succinct suffix arrays based on run-length encoding. *Nordic Journal of Computing*, 12(1):40–66, 2005.
28. G. Navarro. *Compact Data Structures – A Practical Approach*. Cambridge University Press, 2016.
29. D. Okanohara and K. Sadakane. Practical entropy-compressed rank/select dictionary. In *Proc. of 9th Workshop on Algorithm Engineering and Experiments (ALENEX)*, pages 60–70, 2007.
30. Giuseppe Ottaviano and Rossano Venturini. Partitioned elias-fano indexes. In *Proc. of 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 273–282, 2014.
31. M. Pătraşcu and E. Viola. Cell-probe lower bounds for succinct partial sums. In *Proc. 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 117–122, 2010.
32. M. Pătraşcu. Succincter. In *Proc. 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 305–313, 2008.
33. M. Pătraşcu and M. Thorup. Time-space trade-offs for predecessor search. In *Proc. 38th Annual ACM Symposium on Theory of Computing (STOC)*, pages 232–240, 2006.
34. A. R. Quinlan, G. Robins, I. M. Hall, K. Skadron, and R. M. Layer. Binary Interval Search: a scalable algorithm for counting interval intersections. *Bioinformatics*, 29(1):1–7, 11 2012.
35. Naila Rahman and Rajeev Raman. Rank and select operations on binary strings. In Ming-Yang Kao, editor, *Encyclopedia of Algorithms*. Springer, 2008.

36. R. Raman, V. Raman, and S. Rao Satti. Succinct indexable dictionaries with applications to encoding k-ary trees, prefix sums and multisets. *ACM Transactions on Algorithms*, 3(4):43, 2007.
37. K. Sadakane and R. Grossi. Squeezing succinct data structures into entropy bounds. In *Proc. of 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1230–1239, 2006.
38. F. Silvestri. Sorting out the document identifier assignment problem. In *Proc. of 29th European Conference on IR Research (ECIR)*, LNCS 4425, pages 101–112. Springer, 2007.
39. M. D. Soo, R. T. Snodgrass, and C. S. Jensen. Efficient evaluation of the valid-time natural join. In *Proc. 10th International Conference on Data Engineering (ICDE)*, pages 282–292, 1994.