Saving Probe Bits by Cube Domination

Peter Damaschke
Department of Computer Science and Engineering
Chalmers University, 41296 Göteborg, Sweden
ptr@chalmers.se

Abstract

We consider the problem of storing a single element from an m-element set as a binary string of optimal length, and comparing any queried string to the stored string without reading all bits. This is the one-element version of the problem of membership testing in the bit probe model, and solutions can serve as building blocks of general membership testers. Our principal contribution is the equivalence of saving probe bits with some generalized notion of domination in hypercubes. This domination variant requires that every vertex outside the dominating set belongs to a sub-hypercube, of fixed dimension, in which all other vertices belong to in the dominating set. This fixed dimension equals the number of saved probe bits. We give specific constructions showing that up to three probe bits can be ignored when m is far enough from the next larger power of 2. The main technical idea is to use low-dimensional (grid) relaxations of the problem. The design of optimal schemes remains an open problem, however one has to notice that even usual domination in hypercubes is far from being completely understood.

Keywords: bit probe model, dominating set, hypercube

1 Introduction

1.1 The Setting

Consider a fixed set U of m elements. We wish to perform two actions:
(i) Store a single element of U in memory.
(ii) For any $u \in U$ (given to us by some external questioner) answer the question whether the stored element equals u.
Memory may also be empty, and in this case the answer to (ii) should always be negative. An action being similar to (ii) is simply:
(iii) Return the stored element.
One can obviously use (iii) to do (ii), but for a negative answer to (ii) it may not be necessary to identify the stored element.

We suppose that memory consists of some number s of bits. Clearly, the smallest possible memory size for the above task is $s = \lceil \log(m+1) \rceil$, where $\log := \log_2$. Then every element of U can be represented by a unique string of s bits that may be stored. Furthermore it is trivial to recover the stored element by reading all s bits. However it may be sufficient to read a smaller number $t < s$ of bits.

1.2 Contributions

Only at first glance it might appear counterintuitive that not all bits need to be read. However, it suffices to probe a subset of bits that distinguishes the queried element from all other potentially stored elements, and in fact, $t < s$ is possible to achieve if m is far enough from the next larger power of 2. We will see that designing schemes that utilize this idea is equivalent to domination in punctured hypercubes. In a nutshell: Binary strings form a hypercube, but not all vertices need to be used to encode elements, and we can decide on the unused ones. When checking an element, we can stop reading more bits as soon as the already known bits lead to a unique encoding. The vertices described by the read bits form a sub-hypercube. This naturally defines a generalized domination problem in hypercubes that we call cube domination. Since already usual domination in hypercubes is difficult, we relax the problem to low-dimensional grids in order to be able to construct specific solutions. We can save up to 3 probe bits for certain large ranges of m. The main open question is how to prove lower bounds on t, or equivalently, upper bounds on m. Therefore our numerical results are likely to be improved by further research.

Other complexity models and other ways of implementing the same data structure (other than by binary strings) are beyond the scope of this paper.

1.3 Background

A more general task than the one considered here is to store a set of up to n elements and to answer set membership queries, i.e., tell whether a queried element is in the stored set or not. This fundamental data structure problem has been studied thoroughly [5, 15, 4, 16].

The bit-probe model considers the number s of memory bits and the number t of bits that must be read (probed) in order to answer a query. In that model, memory access is assumed to be the expensive part, while the external computations needed to decide and decode the probes are a secondary concern. In general, so called succinct data structures are concerned with the trade-off between memory space and probes for answering queries or executing other data structure operations. There are more recent results on set membership queries in the bit-probe model in different ranges of the parameters [13, 8, 9].

Membership testers for small numbers n of elements can be relevant in various ways. For memory management reasons, a large membership tester may be split into many small blocks, where some hash function determines the blocks where elements are stored. The question of optimizing blocks for fixed small n was raised in [6]. In the present paper we consider only the case $n = 1$. However, testers for single elements can be building blocks of any membership tester that first finds the location in memory where the queried element would be stored, and then checks whether the stored candidate actually equals the queried element. Storing and testing single elements is also a relevant task in its own right, e.g., for authentication and access control of software agents. There, the stored element may be a secret key.

As mentioned, there is a trade-off between s and t. We focus on the smallest possible memory size s per element and ask how many probe bits can be saved nevertheless. In big data applications, memory space can be a bottleneck or an expensive resource.

The immediate relevance of a few saved probe bits might not be large, however for practical s they can make up some percent of the memory bits, and queries may be frequent, such that certain applications may become slightly faster. However the more interesting aspect is an understanding of the combinatorial nature of the problem, and the question which savings are possible at all. Domination is a classic notion in graph theory, and connections between properties of binary codes and the combinatorics of hypercubes are common.

In our notion of cube domination, every vertex not being in the dominating set must belong to some sub-hypercube of fixed dimension, in which all other vertices do belong to the dominating set. Some other, remotely similar generalizations of domination have been considered in the literature. One example are k-dominating sets, where every vertex outside the set must have at least k neighbors in the set. The concept of k-tuple dominating sets is similar. We refer to [1, 7, 11] for several approximation results. In our cube dominating sets, the "dominating" vertices are not all adjacent to the "dominated" ones, instead, they must form some low-diameter subgraph of a fixed shape that includes the dominated vertex. We define our notion only for hypercubes and their subgraphs. Similar concepts of subgraph domination in general graphs might also be interesting, should some motivation for them arise.

2 Preliminaries

To avoid the invention of private terminology we use the established notation for membership testers. Informally, an (m, n, s, t)-scheme is a data structure that can store, within s bits, up to n elements from a universe of m elements and test whether a queried element is among the stored ones, by reading at most t bits. In our case we always have $n = 1$, and hence the "data structure" shall answer question (ii) from the Introduction.

Upon a query, some *strategy* decides on the bits to read and eventually outputs an answer. An *adaptive* strategy decides sequentially which bit to read next, depending on the query and on the bits already seen.

Definition 1. *An* $(m, 1, s, t)$-scheme *consists of:*

- *a partitioning of the set* $\{0,1\}^s$ *of* binary strings *of length* s *into a set* E *of* m element strings*, a set* D *of* dummy strings*, and a set* $\{o\}$ *containing only one special string (without loss of generality the zero string),*
- *an injective function* $\varphi : U \longrightarrow E$ *that encodes the elements of a fixed set* U *with* $|U| = m$ *as element strings,*
- *a deterministic adaptive strategy that takes a string* $e \in E \cup \{o\}$ *and a queried element* $u \in U$*, reads at most* t *bits of* e*, and outputs "yes" if* $e = \varphi(u)$*, and "no" otherwise.*

We may informally identify an element $u \in U$ with its element string $\varphi(u)$. The string o represents the empty set, that is, the state when no element is stored. For our worst-case results, adaptivity is not really needed (see also the later remark after Proposition 3). When checking an element, we expect specific bits at specific positions, hence we can also read these bits simultaneously. However, adaptivity can make a difference for the efficiency of negative tests.

Definition 2. *Given an* $(m, 1, s, t)$*-scheme, we define the following concepts and expressions, where* x_i *denotes the* i*-th symbol of a string* x*.*

- *A* ternary string *is a string* x *from* $\{0, 1, *\}^s$*. We call* $*$ *the* wildcard *and let* $|x|$ *denote the number of non-wildcard symbols.*
- *For* $x, y \in \{0, 1, *\}^s$ *we define* $x \subset y$ *and call* x *a* substring *of* y*,*
  if $\forall i : x_i = y_i \vee x_i = *$*.*

- *A ternary string $p \subset \varphi(u)$ is a* probe *for $u \in U$, if $p \subset e$ is false for all $e \in (E \cup \{o\}) \setminus \{u\}$.*
- *A probe p for u is a* minimal probe *if no ternary string q with $q \subset p$, $q \neq p$ is still a probe for u.*
- *To* read a probe *means to look up the symbols of the probe which are not wildcards.*

Now we obtain a more combinatorial characterization of $(m, 1, s, t)$-schemes and get rid of the consideration of a strategy. Remember that the zero string o represents the empty set and does not need a probe.

Proposition 3. *An $(m, 1, s, t)$-scheme is uniquely determined by φ, E, and some minimal probe $p(u) \subset \varphi(u)$ for every $u \in U$. Moreover, every $p(u)$ contains at least one bit 1. For every $u \in U$, the test whether the stored string is $\varphi(u)$ is done by reading $p(u)$ entirely. Consequently, $t = \max_{u \in U} |p(u)|$.*

Proof. In order to decide the presence of a queried element u, some minimal probe w of u must be read completely (both in the positive and negative case). The reason is that, by minimality, no deviating bit may be found, until the entire probe is read. We give the argument more formally: Let $w' \subset w$ with $|w'| = |w| - 1$. Since w' is not a probe of u, there exists another element $v \in U$ with $w' \subset \varphi(v)$. That is, $\varphi(u)$ and $\varphi(v)$ agree on w' and differ in the bit omitted from w. Without reading this bit, too, a tester cannot tell apart u and v. In particular, the tester cannot safely confirm or rule out the presence of u.

Finally observe that every minimal probe of an element must also contain some 1 in order to distinguish it from the zero string. □

While Definition 1 allowed adaptive reading, the above proof shows that the bits of a minimal probe can be read in any order, even nonadaptively, since we must read them all. In the adaptive setting, reading the probe bits in random order reveals a deviating bit in expected time at most $t/2$, hence negative answers can be given faster, but not positive ones.

Finally, we define $L := \lfloor \log m \rfloor$. Note that $s = L + 1$ is the optimal space that can be used by any $(m, 1, s, t)$-scheme.

3 Cube Domination in Hypercubes

The following graph-theoretic definitions are quite common.

The (Hamming) *weight* of a binary string x is the number of 1s in that string: $|\{i : x_i = 1\}|$. The (Hamming) *distance* between two binary strings x and y of equal lengths is the number of different bit positions: $|\{i : x_i \neq y_i\}|$. The *s-dimensional hypercube Q_s* is the graph with vertex set $\{0,1\}^s$ where two vertices are adjacent if and only if their Hamming distance is 1. The *sub-hypercube* of Q_s described by the ternary string $x \in \{0, 1, *\}^s$ is the subgraph induced by all vertices $y \in \{0,1\}^s$ with $x \subset y$. The *punctured s-dimensional hypercube Q_s^o* is Q_s after removal of one vertex and its indicent edges.

Note that we do not strictly distinguish between elements u, their strings $\varphi(u)$, and their corresponding vertices in the hypercube Q_s; this should not cause confusion. Therefore we may also speak of *element vertices* and *dummy vertices* in Q_s, corresponding to the element and dummy strings, respectively, as introduced in Definition 1.

A subset D of vertices in a graph is called *dominating* if every vertex outside D has a neighbor in D. The *domination number* $\gamma(G)$ of a graph G is the size of a minimum dominating set. We say that a vertex is *dominated* by itself and by its neighbors.

In hypercubes and their subgraphs we will now generalize the notion of domination. As far as we know, this concept is novel. Note that the special case $\ell = 1$ is usual domination in hypercubes.

Definition 4. *In a hypercube Q_s or in an induced subgraph of Q_S, we call a set $D \subseteq \{0,1\}^s$ an* ℓ-cube dominating set *if for every vertex $u \notin D$ there exists an ℓ-dimensional sub-hypercube consisting of u and $2^\ell - 1$ vertices from D. We say that u is* ℓ-cube dominated *by these vertices from D. We call a vertex $v \in D$* redundant *if every $u \notin D$ is already ℓ-cube dominated by some $2^\ell - 1$ vertices from $D \setminus \{v\}$.*

Informally, a redundant vertex is not needed to ℓ-cube dominate any other vertex. The connection of cube domination to our bit-probe number problem is given by:

Proposition 5. *An $(m, 1, L+1, L+1-\ell)$-scheme exists if and only if the vertex set of the punctured $(L+1)$-dimensional hypercube can be divided into a set E of m element vertices and a set D of $2^{L+1}-1-m$ dummy vertices, such that D is an ℓ-cube dominating set. Specifically, a probe for any element $u \in U$ can be chosen as the ternary string of an ℓ-dimensional sub-hypercube which ℓ-cube dominates u.*

Proof. Reformulation of the definition of a probe yields: For any element string u, a substring v with t bits (thereof at least one 1) and $\ell = s - t$ wildcards is a probe, if an only if replacing the wildcards with each of the 2^{s-t} possible combinations of bits generates either u or a dummy string. The assertion is evident from this statement. □

The final lemma in this section is simple but will be a useful tool for concrete constructions of schemes. Techniques for the extension of schemes to larger sizes are also known for general membership testers.

Lemma 6. *Let D be an ℓ-cube dominating set in the full $(L+1)$-dimensional hypercube Q_{L+1} with $|D| = 2^{L+1} - m$, hence with exactly m vertices outside D. Let f be any positive integer. Then we have:*
(i) There exists an $(m-1, 1, L+1, L+1-\ell)$-scheme. If D has some redundant vertex, then there exists an $(m, 1, L+1, L+1-\ell)$-scheme.
(ii) There exists a $(2^f m - 1, 1, L+1+f, L+1+f-\ell)$-scheme. If D has some redundant vertex, then there exists a $(2^f m, 1, L+1+f, L+1+f-\ell)$-scheme.

Proof. (i) We remove some vertex not being in D, to get a punctured hypercube with dummy vertices in D and $m-1$ element vertices. Since no vertex of D has been removed, D is still ℓ-cube dominating, and Proposition 5 yields the first assertion. If some vertex in D is redundant, we remove this vertex instead, and retain m element vertices.

(ii) We append to every string f further bits, in all 2^f possible ways. If the original string was an element (dummy) string, then we let all its 2^f extensions be element (dummy) strings as well. In other words, we glue together 2^f copies of the given hypercube equipped with the given set D. Every element vertex is still ℓ-cube dominated within its own copy, and redundant vertices stay redundant. Now both assertions follow as above. □

4 Ignoring One Bit

Now we use the equivalence to cube domination to obtain concrete $(m, 1, s, t)$-schemes with $s = L+1$ and $t < s$, that is, schemes that use optimal space but need not read all bits. First we deal with $t = L$, that is, $\ell = 1$ and usual dominatiom, according to Proposition 5.

We have to state that the knowledge of exact domination numbers of (full) hypercubes is rather fragmentary. Very few domination numbers are known [2, 3, 10, 12, 14, 17]:

Theorem 7. *The domination number $\gamma(Q_{L+1})$ of the full $(L+1)$-dimensional hypercube is:*

- *7, 12, 16, 32, 62, for $L = 4, 5, 6, 7, 8$, respectively,*
- *2^{L+1-k} for $2^k - 2 \leq L \leq 2^k - 1$,*
- *at most 2^{L-2} for every $L > 6$.*

Let $G - v$ denote the graph G after removal of vertex v and its incident edges. A very simple fact is:

Proposition 8. *For every graph G we have $\gamma(G) - 1 \leq \gamma(G - v) \leq \gamma(G)$.*

Proof. Let D be some minimum dominating set of $G - v$. We re-insert v in $G - v$ to obtain G. If D dominates v, then the domination numbers are equal. If not, it suffices to add v to D to get some dominating set of G. □

Specifically we get the following results to start with.

Proposition 9. *There exists a $(5, 1, 3, 2)$-scheme, but no $(6, 1, 3, 2)$-scheme.*

Proof. We have $\gamma(Q_3^o) = 2$: One example of a minimum dominating set is given by $D = \{100, 011\}$, which yields a $(5, 1, 3, 2)$-scheme that uses the element strings $010, 001, 110, 101, 111$ and the corresponding probes $01*, 0*1, 1*0, 10*, *11$. Due to Proposition 5, a $(6, 1, 3, 2)$-scheme would require a single dominating vertex, which can however dominate only 4 of the 7 vertices. □

Proposition 10. *There exists an $(11, 1, 4, 3)$-scheme, but no $(12, 1, 4, 3)$-scheme.*

Proof. From Theorem 7 we know $\gamma(Q_4) = 4$. Let $o = 0000$ be the vertex that has been deleted to get Q_4^o. Assume that $\gamma(Q_4^o) = 3$, and let D be a dominating set with $|D| = 3$. Since every vertex dominates at most 5 of the 15 vertices of Q_4^o, every vertex must be dominated exactly once. No vertex of weight 1 can be in D, since it would dominate only 4 vertices other than 0000. Hence all vertices of weight 1 must be dominated by vertices of weight 2. Since 1111 must be dominated, too, at most 2 vertices of weight 2 can be in D, say 1100 and 0011. But then no further vertex (of weight 3) can dominate all remaining vertices. Hence $\gamma(Q_4^o) = 4$, contrary to the assumption. Finally, Proposition 5 yields both assertions. □

Similarly, the known domination numbers from Theorem 7 yield a $(24, 1, 5, 4)$-scheme, a $(51, 1, 6, 5)$-scheme, a $(111, 1, 7, 6)$-scheme, and so on, but in these cases we could not figure out whether m can be improved by 1 (which might be possible due to Proposition 8).

However, a more rewarding question than closing these tiny gaps is the bit probe number for general L. Existing knowledge and our preparations enable a very short proof of:

Theorem 11. *For every $L \geq 6$ and every $m < 1.75 \cdot 2^L$, there exists an $(m, 1, L+1, L)$-scheme.*

Proof. By Theorem 7, the $(L+1)$-dimensional hypercube, and thus also the punctured one, has a dominating set of size 2^{L-2}. Due to Proposition 5, the result now follows for $m = 2^{L+1} - 1 - 2^{L-2}$ and for all smaller numbers m. □

An interesting open problem is whether the factor 1.75 can be further increased as L increases. It might even tend to 2 for $L \to \infty$; we do not know a non-trivial bound. According to Theorem 7, this problem is equivalent to the notoriously difficult domination numbers of high-dimensional hypercubes. Therefore we do not investigate this problem further at this point. Instead we look into the opposite direction: We will show that even more probe bits can be saved when the ratios $m/2^L$ are somewhat smaller.

5 Ignoring Two Bits

5.1 Approach

In this section we will construct $(m, 1, L+1, L-1)$-schemes. First we give some intuition.

According to Proposition 5 applied to $\ell = 2$, every element string must belong to a *quadrangle* (2-dimensional sub-hypercube) together with three dummy vertices. On the other hand, since $s = L + 1$, the majority of vertices must be element vertices. To satisfy these seemingly conflicting requirements, any $(m, 1, L+1, L-1)$-scheme needs conglomerates of such quadrangles that share many dummy vertices but include even more different element vertices.

In this situation, our key observation is that a star of dummy vertices, consisting of one "central" vertex and $k \leq L + 1$ of its neighbors, can indeed form quadrangles with up to $\binom{k}{2}$ element vertices at distance 2 from the central vertex. Since $\binom{k}{2} > k + 1$ for $k \geq 4$, the element vertices form the majority. However, the challenge is to pave the entire hypercube by such stars of dummy vertices, thereby maximizing m.

Next, a practical approach to reduce the search space for this puzzle, possibly at the price of getting only suboptimal m, is to map the hypercube to a low-dimensional grid and treat all vertices with the same image equally, i.e., make them either dummy or element vertices. In other words, we relax our problem to some weighted problem in homomorphic images of hypercubes. (Remember that even the usual domination problem in hypercubes is not well understood, hence suboptimal yet "positive" results for $\ell = 2$ should be valuable.)

Below we will develop the technical details of our constructions. First we define several special objects. Recall that $s = L + 1$.

Definition 12. *Let π be a partitioning of the ordered set of bit positions $\{1, \ldots, s\}$ into g* segments *of $s_1, \ldots, s_g$ positions, where $\forall i : s_i > 0$ and $\sum_{i=1}^{g} s_i = s$. Given π, we set up a g-dimensional* grid *of points with integer coordinate vectors $(x_1, \ldots, x_g)$, where $\forall i : 0 \leq x_i \leq s_i$. Furthermore, we partition the set of the $\prod_{i=1}^{g}(s_i+1)$ grid points into a set $\tilde{D}$ of* dummy points *and a set $\tilde{E}$ of* element points*.*
Finally, we define an $(m, 1, s, t)$-scheme from π, $\tilde{D}$, $\tilde{E}$ as follows. We map every vertex v of the punctured s-dimensional hypercube (binary string v of s bits) to the grid point $(x_1, \ldots, x_g)$, where x_i is the number of 1s in the i-th segment of the string v. We let D be the set of strings mapped onto $\tilde{D}$, and similarly with E. That is, every string mapped to a dummy (element) point becomes a dummy (element) string.

In particular, if $g = s$, then the grid is just the original hypercube, and if $g = 1$, then the strings are distinguished by their weights only.

Recall from Proposition 3 that an $(m, 1, s, t)$-scheme is characterized by the set of the element strings and their minimal probes. In Definition 12 we haven't yet specified the probes (and t), but we will do this by cube domination, using Proposition 5, and due to

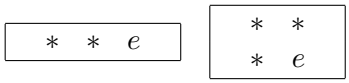
Lemma 6 we consider full hypercubes and redundant vertices in D. In particular, for $\ell = 2$ we will derive a condition on the dummy points in the grid that is sufficient for making the set of dummy elements 2-cube dominating, in the scheme from Definition 12.

For this purrpose, we can alternatively view a grid as a graph where two vertices (grid points) are adjacent if and only if their Euclidean distance is 1. We call a *straight path* a path of grid points where all coordinates but one are equal (figuratively, a path without bends). A *quadrangle* in the grid is a subgraph of four grid points that form a cycle.

Lemma 13. *For any scheme defined by a grid as specified in Definition 12, the following statements are equivalent:*

- *Every element point in the grid is an end point of a straight path with three points, where the two others are dummy points, or belongs to a quadrangle with three dummy points.*
- *The set of dummy vertices is 2-cube dominating in Q_s.*

The two cases are illustrated here, with dummy points and element points displayed as * and e, respectively:



Proof. Consider any quadrangle in the hypercube. Its four strings have the form $u0v0w$, $u0v1w$, $u1v0w$, $u1v1w$, where u, v, w are substrings that may be empty. Remember the notion of segment from Definition 12.

If the two changing bits belong to the same segment, then the four elements are mapped to grid points whose coordinates except one are constant, and the changing coordinate has the values i, $i+1$, $i+1$, $i+2$, respectively, for some integer i.

If the two changing bits belong to different segments, then the four elements are mapped to grid points whose coordinates except two are constant, and the changing coordinates have the values (i, j), $(i+1, j)$, $(i, j+1)$, $(i+1, j+1)$, respectively, for some integers i and j.

Suppose that the set of dummy vertices is 2-cube dominating. Then it follows: Every element vertex is in some quadrangle with three vertices, and the image of this quadrangle is either a straight path with three points or a quadrangle in the grid, as seen above. In the former case, the element vertex is an end point of the path, since the inner point represents two vertices which cannot be one element and one dummy vertex.

This shows one direction of the equivalence. Next we show the more important opposite direction. Consider any element vertex and its image point p in the grid.

Let p be the end of a straight path with dummy points q and r. The strings of vertices mapped to the path differ in only one segment σ. Let the number of 1s in p, q, r in σ be i, $i+1$, $i+2$, respectively. We choose any two positions of bits in σ that are 0 in the considered string mapped to p. Such positions exist since r represents all strings with $i+2$ bits 1 in σ, in particular, the length of σ is at least $i+2$. We change any one of these 0s into 1 to obtain two strings mapped to q, and we change both 0s into 1 to obtain a string mapped to r. These three strings are dummy strings and together 2-cube dominate our considered element string. The case where the strings in p, q, r have $i+2$, $i+1$, i, respectively, bits 1s in σ is symmetric. There we change two 1s into 0s instead.

It should be evident that the reasoning for quadrangles in the grid is similar, except that only one bit is changed in each of two segments. □

5.2 Results

We are ready to derive specific results on $(m, 1, L+1, L-1)$-schemes, using the above tools plus further heuristics (see below).

First we mention that no $(8, 1, 4, 2)$-scheme exists, hence no $(9, 1, 4, 2)$-scheme, etc., exists. This can be shown by tedious exhaustive case examinations, but we have not found a concise proof.

However, for sizes $L \geq 4$ we do obtain $(m, 1, L+1, L-1)$-schemes. Note that we want to maximize m for a given L, and that any $(m, 1, L+1, L-1)$-scheme trivially implies $(m', 1, L+1, L-1)$-schemes for all $m' < m$.

The plan is to place the dummy points, preferably, close to the borders and corners of the grids, because the element numbers are small there. At the same time we must observe the condition from Lemma 13 and "reach" all grid points. In particular, we first try to dominate the central points with largest numbers in the cheapest way.

As a convenient notation, when we display grids, we shall indicate the numbers of strings mapped to each grid point. From Definition 12 it is obvious that these numbers are certain binomial coefficients or products thereof. We also mark each dummy points by an asterisk (*). As we will see, this gives a good overview of the total numbers of dummy and element vertices and makes it easy to check that all element vertices are ℓ-cube dominated.

Proposition 14. *There exists a* $(19, 1, 5, 3)$*-scheme.*

Proof. Apply Lemma 6 (i), Lemma 13, and the following 1-dimensional grid.

$*1$	$*5$	10	10	$*5$	$*1$

We remark that we get only $m = 19$ rather than 20, due to the lack of a redundant dummy vertex. □

Corollary 15. *There exist a* $(39, 1, 6, 4)$*-scheme, a* $(79, 1, 7, 5)$*-scheme, a* $(159, 1, 8, 6)$*-scheme, and a* $(319, 1, 9, 7)$*-scheme.*

Proof. Apply Lemma 6 (ii) to the scheme from Proposition 14. □

The next item in this sequence would be a $(639, 1, 10, 8)$-scheme, but this can be considerably improved:

Proposition 16. *There exists a* $(723, 1, 10, 8)$*-scheme.*

Proof. Apply Lemma 6 (i), Lemma 13, and the following 2-dimensional grid.

1	$*5$	$*10$	$*10$	5	1
5	25	$*50$	50	25	$*5$
$*10$	50	100	100	$*50$	$*10$
$*10$	$*50$	100	100	50	$*10$
$*5$	25	50	$*50$	25	5
1	5	$*10$	$*10$	$*5$	1

Indeed, all element points satisfy the condition in Lemma 13. □

As the next step we may apply Lemma 6 (ii) once more and obtain a $(1447, 1, 11, 9)$-scheme. At this point we can raise m somewhat. More interesting than the relatively small improvement as such is the fact that a new pattern of dummy points emerges in the grid.

Proposition 17. *There exists a* $(1464, 1, 11, 9)$*-scheme.*

Proof. Apply Lemma 6 (i), Lemma 13, and the following 2-dimensional grid.

$*1$	6	$*15$	$*20$	$*15$	6	$*1$
5	30	75	$*100$	75	30	5
$*10$	$*60$	150	200	150	$*60$	$*10$
$*10$	$*60$	150	200	150	$*60$	$*10$
5	30	75	$*100$	75	30	5
$*1$	6	$*15$	$*20$	$*15$	6	$*1$

Note as a minor detail that the dummy vertices at the corners are redundant, so we can remove one of them. They are dummy points in our scheme, only because they cannot be 2-cube dominated cheaply. □

Finally we get a general result for saving two probe bits:

Theorem 18. *For every* $L \geq 10$ *and every* $m < 1.42 \cdot 2^L$*, there exists an* $(m, 1, L + 1, L - 1)$*-scheme.*

Proof. This follows instantly from Lemma 6 and Proposition 17, noticing that $1464/1024 > 1.42$. □

Similarly to the remarks on Theorem 11, the factor 1.42 is probably not the last word. We conjecture that it can be raised further. But we mention that we have not found better values of m via grids of dimension $g > 2$ so far. An explanation might be that the quadrangles themselves are 2-dimensional grids, such that higher-dimensional grids might not give better opportunities to glue them.

5.3 Ignoring Even More Bits

Next we can show that even some $(m, 1, L + 1, L - 2)$-schemes exist. Similarly as in case $\ell = 2$, the condition for a scheme described by a 2-dimensional grid is that every element point is the end of a straight path of four points (a (1×4)-subgrid) or the corner of some (2×3)-subgrid, where all other points in the respective subgrid are dummy points. The proof is analogous to Lemma 13.

Theorem 19. *For every* $L \geq 7$ *and every* $m < 1.09 \cdot 2^L$*, there exists an* $(m, 1, L + 1, L - 2)$*-scheme.*

Proof. Our basic observation is a 1-dimensional grid describing a $(69, 1, 7, 4)$-scheme (where we remove one of the $35 + 35$ element vertices in the punctured hypercube). The grid is:

$*1$	$*7$	$*21$	35	35	$*21$	$*7$	$*1$

Note that every element point is the end of a path with three dummy points. Since $70/64 > 1.09$, Lemma 6 implies the assertion for every larger L. □

Again, it could be interesting to raise the factor 1.09. We conclude with a

Conjecture. There exist no $(m, 1, L + 1, L - 3)$-schemes.

We were led to this conjecture by reasoning on the growth of binomial coefficients, and from failed attempts to construct such schemes from grids despite much experimentation. But actually it is open whether $(m, 1, L + 1, L + 1 - \ell)$-schemes are possible even for arbitrarily large ℓ. A weaker conjecture is that no such schemes exist from some fixed ℓ on.

6 Conclusions

We have constructed several non-obvious schemes for storing one out of m elements, that use optimal space and save some probe bits when querying the stored element. They work for large ranges of m. Although being based on fairly general ideas (cube domination in hypercubes, low-dimensional relaxation, and doubling), the specific results have been found in an ad-hoc way. It could be nice to find good general patterns of dummy points.

The impact of extra memory bits ($s = L + 2$, $s = L + 3$, etc.) was beyond our scope. Of course, larger probe bit savings would not be surprising there, but it remains the (difficult) question of exactly quantifying them. Some other open questions were already mentioned. Finally, these schemes for single elements could be used inside membership testers for subsets of up to n elements.

Acknowledgment

Special thanks go to the anonymous reviewer who pointed out additional references around Theorem 7.

References

[1] Argiroffo, G.R., Leoni, V., Torres, P.: On the Complexity of k-Domination and k-Tuple Domination in Graphs. Info. Proc. Letters 115, 556–561 (2015)

[2] Arumugam, S., Kala, R.: Domination Parameters of Hypercubes. J. Indian Math. Soc. 65, 31–38 (1998)

[3] Azarija, J., Henning, M.A., Klavzar, S.: (Total) Domination in Prisms. Electron. J. Combin. 24, paper 1.19 (2017)

[4] Buhrman, H., Miltersen, P.B., Radhakrishnan, J., Venkatesh, S.: Are Bitvectors Optimal? SIAM J. Comput. 31, 1723–1744 (2002)

[5] Carter, L., Floyd, R.W., Gill, J., Markowsky, G., Wegman, M.N.: Exact and Approximate Membership Testers. In: Lipton, R.J. et al. (Eds.) STOC 1978. ACM, pp. 59–65 (1978)

[6] Fan, B., Andersen, D.G., Kaminsky, M., Mitzenmacher, M.: Cuckoo Filter: Practically Better than Bloom. In: Seneviratne, A. et al. (Eds.) CoNEXT 2014. ACM, pp. 75–88 (2014)

[7] Förster, K.T.: Approximating Fault-Tolerant Domination in General Graphs. In: Nebel, M.E., Szpankowski, W. (Eds.) ANALCO 2013. SIAM, pp. 25–32 (2013)

[8] Garg, M., Radhakrishnan, J.: Set Membership With a Few Bit Probes. In: Indyk, P. (Ed.) SODA 2015. ACM-SIAM, pp. 776–784 (2015)

[9] Garg, M., Radhakrishnan, J.: Set Membership With Non-Adaptive Bit Probes. In: Vollmer, H., Vallée, B, (Eds.) STACS 2017. LIPIcs, vol. 66, paper 38, Dagstuhl (2017)

[10] Harary, F., Livingston, M.: Independent Domination in Hypercubes. Appl. Math. Letters 6, 27–28 (1993)

[11] Klasing, R., Laforest, C.: Hardness Results and Approximation Algorithms of k-Tuple Domination in Graphs. Info. Proc. Letters 89, 75–83 (2004)

[12] Klavzar, S., Ma, M.: The Domination Number of Exchanged Hypercubes. Info. Proc. Letters 114, 159–162 (2014)

[13] Lewenstein, M., Munro, J.I., Nicholson, P.K., Raman, V.: Improved Explicit Data Structures in the Bitprobe Model. In: Schulz, A.S., Wagner, D. (Eds.) ESA 2014. LNCS, vol. 8737, pp. 630–641, Springer, Heidelberg (2014)

[14] Östergård, P.R.J., Blass, U.: On the Size of Optimal Binary Codes of Length 9 and Covering Radius 1. IEEE Trans. Inform. Theory 47, 2556–2557 (2001)

[15] Pagh, R.: On the Cell Probe Complexity of Membership and Perfect Hashing. In: Vitter, J.S., Spirakis, P.G., Yannakakis, M. (eds.) STOC 2001. ACM, pp. 425–432 (2001)

[16] Raman, R., Raman, V., Satti, S.R.: Succinct Indexable Dictionaries with Applications Encoding k-ary Trees, Prefix Sums and Multisets. ACM Trans. Algor. 3, paper 43 (2007)

[17] van Wee, G.J.M.: Improved Sphere Bounds on the Covering Radius of Codes. IEEE Trans. Inform. Theory 34, 237–245 (1988)