Lower Bounds for Dynamic Connectivity (2004; Pătraşcu, Demaine)

Mihai Pătraşcu, MIT, `web.mit.edu/~mip/www/`

Index terms: partial-sums problem, prefix sums, dynamic lower bounds
Synonyms: dynamic trees

1 Problem Definition

The dynamic connectivity problem asks to maintain a graph G subject to the following operations:

`insert`(u, v)**:** insert an undirected edge (u, v) into the graph.
`delete`(u, v)**:** delete the edge (u, v) from the graph.
`connected`(u, v)**:** test whether u and v lie in the same connected component.

Let m be an upper bound on the number of edges in the graph. This entry discusses cell-probe lower bounds for this problem. Let t_u be the complexity of `insert` and `delete`, and t_q be the complexity of `query`.

The partial-sums problem. Lower bounds for dynamic connectivity are intimately related to lower bounds for another classic problem: maintaining partial sums. Formally, the problem asks to maintain an array $A[1\,..\,n]$ subject to the following operations:

`update`(k, Δ)**:** let $A[k] \leftarrow \Delta$.
`sum`(k)**:** returns the partial sum $\sum_{i=1}^{k} A[i]$.
`testsum`(k, σ)**:** returns a boolean value indicating whether `sum`$(k) = \sigma$.

To specify the problem completely, let elements $A[i]$ come from an arbitrary group G, containing at least 2^δ elements. In the cell-probe model with b-bit cells, let t_u^Σ be the complexity of `update`, and t_q^Σ be the complexity of `testsum` (which is also a lower bound on `sum`).

The trade-offs between t_u^Σ and t_q^Σ are well understood for all values of b and δ. However, this entry only considers lower bounds under the standard assumptions that $b = \Omega(\lg n)$ and $t_u \geq t_q$. Assuming $b = \Omega(\lg n)$ is standard for upper bounds in the RAM model, and also means that the lower bound applies to the pointer machine. Then, Pătraşcu and Demaine [6] prove:

Theorem 1. *The complexity of the partial-sums problems satisfies:* $t_q^\Sigma \cdot \lg(t_u^\Sigma / t_q^\Sigma) = \Omega(\frac{\delta}{b} \cdot \lg n)$.

Observe that this matches the text-book upper bound using augmented trees. One can build a balanced binary tree over $A[1], \ldots, A[n]$, and store in every internal node the sum of its subtree. Then, updates and queries touch $O(\lg n)$ nodes (and spend $O(\lceil \delta/b \rceil)$ time in each one, due to the size of the group). To decrease the query time, one can use a B-tree.

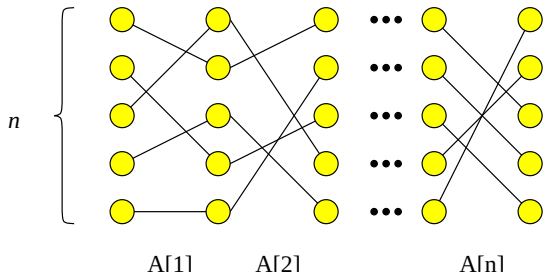

Figure 1: Constructing an instance of dynamic connectivity that mimics the partial-sums problem.

Relation to dynamic connectivity. We now clarify how lower bounds for maintaining partial sums imply lower bounds for dynamic connectivity. Consider the partial-sums problem over the group $G = S_n$, i.e., the permutation group on n elements. Note that $\delta = \lg(n!) = \Omega(n \lg n)$. It is standard to set $b = \Theta(\lg n)$, as this is the natural word size used by dynamic connectivity upper bounds. This implies $t_q^\Sigma \lg(t_u^\Sigma / t_q^\Sigma) = \Omega(n \lg n)$.

The lower bound follows from implementing the partial-sums operations using dynamic connectivity operations. Refer to Figure 1. The vertices of the graph form an integer grid of size $n \times n$. Each vertex is incident to at most two edges, one edge connecting to a vertex in the previous column and one edge connecting to a vertex in the next column. Point (x, y_1) in the grid is connected to point $(x+1, A[x](y_1))$, i.e. the edges between two adjacent columns describe the corresponding permutation from the partial sums vector.

To implement `update`(x, π), all the edges between column x and $x+1$ are first deleted, and then new edges are inserted according to π. This gives $t_u^\Sigma = O(2n \cdot t_u)$. To implement `testsum`$(x, \pi)$, one can use n `connected` queries, between the pairs of points $(1, y) \rightsquigarrow (x+1, \pi(y))$. Then, $t_q^\Sigma = O(n \cdot t_q)$. Observe that the `sum` query cannot be implemented as easily. Dynamic connectivity is the main motivation to study the `testsum` query.

The lower bound of Theorem 1 translates into $nt_q \cdot \lg \frac{2nt_u}{nt_q} = \Omega(n \lg n)$, hence $t_q \lg \frac{t_u}{t_q} = \Omega(\lg n)$. Note that this lower bound implies $\max\{t_u, t_q\} = \Omega(\lg n)$. The best known upper bound (using amortization and randomization) is $O(\lg n (\lg \lg n)^3)$ [9]. For any $t_u = \Omega(\lg n (\lg \lg n)^3)$, the lower bound trade-off is known to be tight. Note that the graph in the lower bound is always a disjoint union of paths. This implies optimal lower bounds for two important special cases: dynamic trees [8], and dynamic connectivity in plane graphs [2].

2 Key Results

2.1 Understanding Hierarchies

Epochs. To describe the techniques involved in the lower bounds, first consider the `sum` query, and assume $\delta = b$. In 1989, Fredman and Saks [3] initiated the study of dynamic cell-probe lower bounds, essentially showing a lower bound of $t_q^\Sigma \lg t_u^\Sigma = \Omega(\lg n)$. Note that this implies $\max\{t_q^\Sigma, t_u^\Sigma\} = \Omega(\lg n / \lg \lg n)$.

At an intuitive level, their argument proceeded as follows. The hard instance will have n random updates, followed by one random query. Let $r \geq 2$ to be determined. Looking *back* in time from the query, one groups the updates into exponentially growing epochs: the latest r updates are epoch

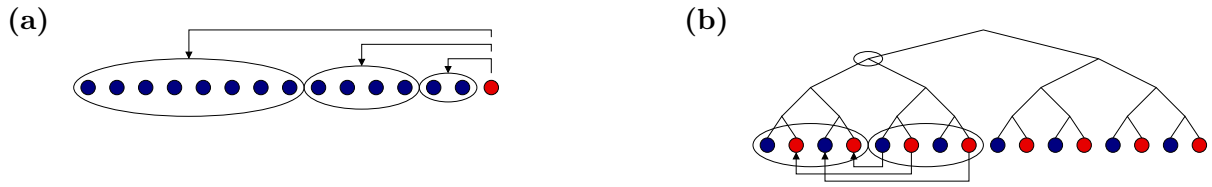

Figure 2: Analysis of cell probes in the (a) epochs-based, and (b) time hierarchy, techniques.

1, the earlier r^2 updates are epoch 2 etc. Note that epoch numbers increase going back in time, and there are $O(\log_r n)$ epochs in total.

For some epoch i, consider revealing to the query all updates performed in all epochs different from i. Then, the query reduces to a partial-sums query among the updates in epoch i. Unless the query is to an index below the minimum index updated in epoch i, the answer to the query is still uniformly random, i.e. has δ bits of entropy. Furthermore, even if one is given, say, $r^i\delta/100$ bits of information about epoch i, the answer still has $\Omega(\delta)$ bits of entropy on average. This is because the query and updates in epoch i are uniformly random, so the query can ask for any partial sum of these updates, uniformly at random. Each of the r^i partial sums is an independent random variable of entropy δ.

Now one can ask how much information is available to the query. At the time of the query, let each cell be associated with the epoch during which it was last written. Choosing an epoch i uniformly at random, one can make the following intuitive argument:

1. any cells written by epochs $i+1, i+2, \ldots$ cannot contain information about epoch i, as they were written in the past.
2. in epochs $1, \ldots, i-1$ a number of $bt_u^\Sigma \cdot \sum_{j=1}^{i-1} r^j \leq bt_u^\Sigma \cdot 2r^{i-1}$ bits were written. This is less than $r^i\delta/100$ bits of information for $r > 200t_u^\Sigma$ (recall the assumption $\delta = b$). By the above, this implies the query answer still has $\Omega(\delta)$ bits of entropy.
3. since i is uniformly random among $\Theta(\log_r n)$ epochs, the query makes an expected $O(t_q^\Sigma/\log_r n)$ probes to cells from epoch i. All queries which make no cell probes to epoch i have a fixed answer (entropy 0), and all other queries have answers of entropy $\leq \delta$. Since an average query has entropy $\Omega(\delta)$, a query must probe a cell from epoch i with constant probability. That means $t_q^\Sigma/\log_r n = \Omega(1)$, and $t^\Sigma = \Omega(\log_r n) = \Omega(\lg n/\lg t_u^\Sigma)$.

One should appreciate the duality between the proof technique and the natural upper bounds based on a hierarchy. Consider an upper bound based on a tree of degree r. The last r random updates (epoch 1) are likely to be uniformly spread in the array. This means the updates touch different children of the root. Similarly, the r^2 updates in epoch 2 are likely to touch every node on level 2 of the tree, and so on. Now, the lower bound argues that the query needs to traverse a root-to-leaf path, probing a node on every level of the tree (this is equivalent to one cell from every epoch).

Time hierarchies. Despite considerable refinement to the lower bound techniques, the lower bound of $\Omega(\lg n/\lg\lg n)$ was not improved until 2004. Then, Pătraşcu and Demaine showed an optimal bound of $t_q^\Sigma \lg(t_u^\Sigma/t_q^\Sigma) = \Omega(\lg n)$, implying $\max\{t_u^\Sigma, t_q^\Sigma\} = \Omega(\lg n)$. For simplicity, the discussion below disregards the trade-off, and just sketches the $\Omega(\lg n)$ lower bound.

Their counting technique is rather different from the epoch technique; refer to Figure 2. The hard instance is a sequence of n operations alternating between updates and queries. They consider a balanced binary tree over the time axis, with every leaf being an operation. Now for every node

of the tree, they propose to count the number of cell probes made in the right subtree, to a cell written in the left subtree. Every probe is counted exactly once, for the lowest common ancestor of the read and write times.

Now focus on two sibling subtrees, each containing k operations. The $k/2$ updates in the left subtree, and the $k/2$ queries in the right subtree, are expected to interleave in index space. Thus, the queries in the right subtree ask for $\Omega(k)$ different partial sums of the updates in the left subtree. Thus, the right subtree "needs" $\Omega(k\delta)$ bits of information about the left subtree, and this information can only come from cells written in the left subtree and read in the right one. This implies a lower bound of $\Omega(k)$ probes, associated with the parent of the sibling subtrees. This bound is linear in the number of leaves, so summing up over the tree, one obtains a total $\Omega(n \lg n)$ lower bound, or $\Omega(\lg n)$ cost per operation.

An optimal epoch construction. Rather surprisingly, Pătraşcu and Tarniţă [7] managed to reprove the optimal trade-off of Theorem 1 with minimal modifications to the epoch argument. In the old epoch argument, the information revealed by epochs $1, \ldots, i-1$ about epoch i was bounded by the number of cells written in these epochs. The key idea is that an equally good bound is the number of cells read during epochs $1, \ldots, i-1$ and written during epoch i.

In principle, all cell reads from epoch $i-1$ could read data from epoch i, making these two bounds identical. However, one can randomize the epoch construction, by inserting the query after an unpredictable number of updates. This randomization "smoothes" out the distribution of epochs from which cells are read, i.e. a query reads $O(t_q^\Sigma / \log_r n)$ cells from every epoch, in expectation over the randomness in the epoch construction. Then, the $O(r^{i-1})$ updates in epochs $1, \ldots, i-1$ only read $O(r^{i-1} \cdot t_u^\Sigma / \log_r n)$ cells from epoch i. This is not enough information if $r \gg t_u^\Sigma / \log_r n = \Theta(t_u^\Sigma / t_q^\Sigma)$, which implies $t_q^\Sigma = \Omega(\log_r n) = \Omega(\lg n / \lg \frac{t_u^\Sigma}{t_q^\Sigma})$.

2.2 Technical Difficulties

Nondeterminism. The lower bounds sketched above are based on the fact that the `sum` query needs to output $\Omega(\delta)$ bits of information about every query. If dealing with the *decision* `testsum` query, an argument based on output entropy can no longer work.

The most successful idea for decision queries has been to convert them to queries with non-boolean output, in an extended cell-probe model that allows nondeterminism. In this model, the query algorithm is allowed to spawn an arbitrary number of computation threads. Each thread can make t_q cell probes, after which it must either reject, or return an answer to the query. All non-rejecting threads must return the same output. In this model, a query with arbitrary output is equivalent to a decision query, because one can just nondeterministically guess the answer, and then verify it.

By the above, the challenge is to prove good lower bounds for `sum` even in the nondeterminstic model. Nondeterminism shakes our view that when analyzing epoch i, only cell probes to epoch i matter. The trouble is that the query may not know *which* of its probes are actually to epoch i. A probe that reads a cell from a previous epoch provides at least some information about epoch i: no update in the epoch decided to overwrite the cell. Before this was not a problem, because the goal was only to rule out the case that there are *zero* probes to epoch i. Now, however, different threads can probe any cell in memory, and one cannot determine which threads actually avoid probing anything in epoch i. In other words, there is a covert communication channel between epoch i and

the query, in which the epoch can use the choice of which cells to write in order to communicate information to the query.

There are two main strategies for handling nondeterministic query algorithms. Husfeldt and Rauhe [4] give a proof based on some interesting observations about the combinatorics of nondeterministic queries. Pătraşcu and Demaine [6] use the power of nondeterminism itself, to output a small certificate that rules out useless cell probes. The latter result implies the optimal lower bound of Theorem 1 for `testsum` and, thus, the logarithmic lower bound for dynamic connectivity.

Alternative histories. The framework described above relies on fixing all updates in epochs different from i to an average value, and arguing that the query answer still has a lot of variability, depending on updates in epoch i. This is true for aggregation problems, but not for search problems. If a searched item is found with equal probability in any epoch, then fixing all other epochs renders epoch i irrelevant with probability $1 - \frac{1}{\log_r n}$.

Alstrup, Husfeldt and Rauhe [1] propose a very interesting refinement to the technique, proving $\Omega(\lg n / \lg\lg n)$ lower bounds for an impressive collection of search problems. Intuitively, their idea is to consider $O(\log_r n)$ alternative histories of updates, chosen independently at random. Epoch i is relevant in at least one of the histories with constant probability. On the other hand, even if one knows what epochs 1 through $i-1$ learned about epoch i in *all histories*, answering a random query is still hard.

Bit-probe complexity. Intuitively, if the word size is $b = 1$, the lower bound for connectivity should be roughly $\Omega(\lg^2 n)$, because a query needs $\Omega(\lg n)$ bits from every epoch. However, ruling out anything except zero probes to an epoch turns out to be difficult, for the same reason that the nondeterministic case is difficult. Without giving a very satisfactory understanding of this issue, Pătraşcu and Tarniţă [7] use a large bag of tricks to show an $\Omega((\frac{\lg n}{\lg\lg n})^2)$ lower bound for dynamic connectivity. Furthermore, they consider the partial-sums problem in $\mathbb{Z}_2$, and show an $\Omega(\frac{\lg n}{\lg\lg\lg n})$ lower bound, which is a triply-logarithmic factor away from the upper bound!

3 Applications

The lower bound discussed here extends by easy reductions to virtually all natural fully-dynamic graph problems [6].

4 Open Problems

By far, the most important challenge for future research is to obtain a lower bound of $\omega(\lg n)$ per operation for some dynamic data structure in the cell-probe model with word size $\Theta(\lg n)$. Miltersen [5] specifies a set of technical conditions for what qualifies as a solution to such a challenge. In particular, the problem should be a *dynamic language membership* problem.

For the partial-sums problem, though `sum` is perfectly understood, `testsum` still lacks tight bounds for certain ranges of parameters [6]. In addition, obtaining tight bounds in the bit-probe model for partial sums in $\mathbb{Z}_2$ appears to be rather challenging.

5 Experimental Results

None is reported.

6 Data Sets

None is reported.

7 URL to Code

None is reported.

8 Cross References

Please link to sections on dynamic connectivity upper bounds.

9 Recommended Reading

[1] S. Alstrup, T. Husfeldt, and T. Rauhe, *Marked ancestor problems*, in Proc. 39th IEEE Symposium on Foundations of Computer Science (FOCS), 1998, pp. 534–543.

[2] D. Eppstein, G. F. Italiano, R. Tamassia, R. E. Tarjan, J. R. Westbrook, and M. Yung, *Maintenance of a minimum spanning forest in a dynamic planar graph*, Journal of Algorithms, 13 (1992), pp. 33–54. See also SODA'90.

[3] M. L. Fredman and M. E. Saks, *The cell probe complexity of dynamic data structures*, in Proc. 21st ACM Symposium on Theory of Computing (STOC), 1989, pp. 345–354.

[4] T. Husfeldt and T. Rauhe, *New lower bound techniques for dynamic partial sums and related problems*, SIAM Journal on Computing, 32 (2003), pp. 736–753. See also ICALP'98.

[5] P. B. Miltersen, *Cell probe complexity - a survey*, in 19th Conference on the Foundations of Software Technology and Theoretical Computer Science (FSTTCS), 1999. Advances in Data Structures Workshop.

[6] M. Pătraşcu and E. D. Demaine, *Logarithmic lower bounds in the cell-probe model*, SIAM Journal on Computing, 35 (2006), pp. 932–963. See also SODA'04 and STOC'04.

[7] M. Pătraşcu and C. Tarniţă, *On dynamic bit-probe complexity*, Theoretical Computer Science, (2007). To appear. See also ICALP'05.

[8] D. D. Sleator and R. E. Tarjan, *A data structure for dynamic trees*, Journal of Computer and System Sciences, 26 (1983), pp. 362–391. See also STOC'81.

[9] M. Thorup, *Near-optimal fully-dynamic graph connectivity*, in Proc. 32nd ACM Symposium on Theory of Computing (STOC), 2000, pp. 343–350.