Matching Nuts and Bolts Optimally[*]

Phillip G. Bradford

Max-Planck-Institut für Informatik
Im Stadtwald
66123 Saarbrücken, Germany.

E-mail: bradford@mpi-sb.mpg.de.

September 1995

Abstract

The nuts and bolts problem is the following : Given a collection of n nuts of distinct sizes and n bolts of distinct sizes such that for each nut there is exactly one matching bolt, find for each nut its corresponding bolt subject to the restriction that we can *only* compare nuts to bolts. That is we can neither compare nuts to nuts, nor bolts to bolts. This humble restriction on the comparisons appears to make this problem quite difficult to solve. In this paper, we illustrate the existence of an algorithm for solving the nuts and bolts problem that makes $O(n \lg n)$ nut-and-bolt comparisons. We show the existence of this algorithm by showing the existence of certain expander-based comparator networks. Our algorithm is asymptotically optimal in terms of the number of nut-and-bolt comparisons it does. Another view of this result is that we show the existence of a decision tree with depth $O(n \lg n)$ that solves this problem.

1 Introduction

In [20], page 293, Rawlins posed the following interesting problem :

> *We wish to sort a bag of n nuts and n bolts by size in the dark. We can compare the sizes of a nut and a bolt by attempting to screw one into the other. This operation tells us that either the nut is bigger than the bolt; the bolt is bigger than the nut; or they are the same size (and so fit together). Because it is dark we are not allowed to compare nuts directly or bolts directly.*
>
> *How many fitting operations do we need to sort the nuts and bolts in the worst case?*

As a computer scientist (instead of a carpenter) you might prefer to see the problem stated as follows (Alon *et al.* [6]) :

> *Given two sets $B = \{b_1, \ldots, b_n\}$ and $S = \{s_1, \ldots, s_n\}$, where B is a set of n distinct real numbers (representing the sizes of the bolts) and S is a permutation of B, we wish to find efficiently the unique permutation $\sigma \in S_n$ so that $b_i = s_{\sigma(i)}$ for all i, based on queries of the form* compare b_i and s_j. *The answer to each such query is either $b_i > s_j$ or $b_i = s_j$ or $b_i < s_j$.*

[*]The author was supported by the ESPRIT Basic Research Actions Program, under contract No. 7141 (project ALCOM II).

The obvious information theoretic lower bound shows that at least $\Omega(n \lg n)$ nut-and-bolt comparisons are needed to solve the problem, even for a randomized algorithm. In fact, there is a simple randomized algorithm which achieves an expected running time of $O(n \lg n)$, namely Quicksort : Pick a random nut, find its matching bolt, and then split the problem into two subproblems which can be solved recursively, one consisting of the nuts and bolts smaller than the matched pair and one consisting of the larger ones. The standard analysis of randomized Quicksort gives the expected running time as stated above (see for example [8, 20]).

Unfortunately, it seems much harder to find an efficient deterministic algorithm. The first $O(n \lg^{O(1)} n)$-time deterministic algorithm was by Alon *et al.* [6] which is also based on Quicksort and takes $\Theta(n \lg^4 n)$ time. They mention in passing that they also have an $O(n \lg^{3+\epsilon} n)$ time algorithm for any $\epsilon > 0$. To find a good pivot element which splits the problem into two subproblems of nearly the same size, they run $\lg n$ iterations of a procedure which eliminates half of the nuts in each iteration while maintaining at least one good pivot; since there is only one nut left in the end, this one must be a good pivot. This procedure uses the edges of an efficient expander of degree $\Theta(\lg^2 n)$ to define its comparisons. Therefore, finding a good pivot takes $\Theta(n \lg^3 n)$ time, and the entire Quicksort takes $\Theta(n \lg^4 n)$ time.

Bradford and Fleischer [7] give a very simple $O(n \lg^2 n)$-time algorithm by building an $O(n \lg n)$-time algorithm for pivot selection which uses explicitly constructed expanders. However, initially the constants of Bradford and Fleischer's algorithm were worse than those of Alon *et al.*'s algorithm since Bradford and Fleischer iteratively construct an expander with suitable parameters from simpler expanders. Later on, Alon suggested a simple way to improve the constants considerably. While working on a draft of this paper we learned that Komlós, Ma, and Szemerédi also have an $O(n \lg n)$-time algorithm for solving the nuts and bolts problem [12].

In this paper, we show the existence of an asymptotically optimal algorithm (in terms of nut-and-bolt comparisons) to find a good pivot. We do this by showing that comparator networks that are ϵ-halvers exist for nuts and bolts. An ϵ-halver approximately splits a set of n elements with $O(n)$ work. This approximate splitting is enough to allow us to select good pivots while iterating ϵ-halvers on geometrically smaller sets of nuts and bolts. The hard part in building these ϵ-halvers is to ensure that nuts are never compared to nuts and bolts are never compared to bolts while maintaining the ϵ-halving property. In these ϵ-halvers we must account for both the errors and the loss of comparable elements and this takes the bulk of the paper. In some sense this is somewhat reminiscent of Paterson's version [16] of the famous Ajtai, Komlós, and Szemerédi [2, 3] sorting network. Although we are not working under such time constraints in parallel.

We show that there is a "good" pivot selection algorithm using only $O(n)$ nut-and-bolt comparisons which leads directly to the existence of our $O(n \lg n)$ nut-and-bolt comparison algorithm. Our algorithm is asymptotically optimal in terms of the number of nut-and-bolt comparisons it does. We remark that it is not uncommon for papers to show the existence of algorithms with a desirable number of element comparisons, but where the determination of the choices of which comparisons to make is more expensive, see for example [1, 2, 3, 5, 16, 19].

Alon *et al.* [6] mention two potential applications of the nuts and bolts problem: the first is local sorting of nodes in a given graph [10], and the second is selection of read only memory with a little read/write memory [14].

In the next section, we describe the Quicksort algorithm more formally and recall some facts about expanders. In Section 3, we show how we can efficiently find a good pivot with $O(n)$ nut-and-bolt comparisons. We conclude with some remarks in Section 4.

2 Basic Definitions

Let $S = \{s_1, \dots, s_n\}$ be a set of nuts of different sizes and $B = \{b_1, \dots, b_n\}$ be a set of corresponding bolts. For a nut $s \in S$ define $rank(s)$ as $|\{t \in B \mid s \geq t\}|$. The rank of a bolt is defined similarly. For a constant $c < \frac{1}{2}$, s is called a *c-approximate median* if $cn \leq rank(s) \leq (1-c)n$. Similarly, define the *relative rank* of s with respect to a subset $T \subseteq B$ as $rank_T(s) := \dfrac{|\{t \in T \mid s \geq t\}|}{|T|}$.

The algorithm for matching nuts and bolts works as follows.

(1) Find a c-approximate median s of the n given nuts (we will determine c later).

(2) Find the bolt b corresponding to s.

(3) Compare all nuts to b and all bolts to s. This gives two piles of nuts (and bolts as well), one with the nuts (bolts) smaller than s and one with the nuts (bolts) bigger than s.

(4) Run the algorithm recursively on the two piles of the smaller nuts and bolts and the two piles of the bigger nuts and bolts.

In the next section, we will show how to find a c-approximate median with $O(n)$ nut-and-bolt comparisons, where c is a constant. Then our main result follows immediately.

Theorem 1 We can match n nuts with their corresponding bolts in $O(n \lg n)$ nut-and-bolt comparisons.

Proof: The correctness of the algorithm above follows immediately from the correctness of Quicksort. For the running time in terms of nut-and-bolt comparisons observe that each subproblem has size at most $(1-c)n$, hence the depth of the recursion is only $O(\lg n)$, and in each level of the recursion the total number of nut-and-bolt comparisons to get all of the c-approximate medians is $O(n)$. ∎

We now recall some facts about expanders (see for example [13] if you want to learn more about expanders). An (n, d, c)*-expander* is a d-regular bipartite graph on vertices $\mathcal{I}$ (inputs) and $\mathcal{O}$ (outputs), where $|\mathcal{I}| = |\mathcal{O}| = n$, such that every subset $A \subseteq \mathcal{I}$ that contains up to $n/2$ elements, is joined by edges to at least $|A| \left(1 + c(1 - \frac{|A|}{n})\right)$ different outputs. The constant c is called the *expansion factor* of the graph. Further, we will always take the degree d to be constant, albeit very large. A *strong* (n, d, c)-expander is a d-regular bipartite graph on vertices $\mathcal{I}$ (inputs) and $\mathcal{O}$ (outputs), where $|\mathcal{I}| = |\mathcal{O}| = n$, such that *every* subset $A \subseteq \mathcal{I}$ is joined by edges to at least $|A| \left(1 + c(1 - \frac{|A|}{n})\right)$ different outputs, see Alon [4].

Lemma 1 (Alon [4], Lemma 3.2) Any $(2n, d, c)$ expander (expanding from subsets of both the inputs and outputs) is a (n, d, b)-strong expander, where $b = 2c/\left((d+1)(c+1)\right)$.

It is not hard to show the existence of expanders, see Sarnak's book [21] or Lubotzky's book [13] and their citations. On the other hand, it appears to be much more difficult to explicitly construct expanders with provably good expansion factors. Although several researchers have given explicit constructions of expanders with provably good expansion factors.

The proof of the next corollary follows from the standard literature on graph expanders.

Corollary 1 Let $0 < \alpha \leq \beta < 1$ be constants and $\gamma_{(\alpha,\beta)} = \frac{\frac{\beta}{\alpha} - 1}{\alpha - 1}$. Then there exists an integer $d_{(\alpha,\beta)}$ such that we can construct a strong $(n, d_{(\alpha,\beta)}, \gamma_{(\alpha,\beta)})$-expander in $O(n)$ time, where any subset of the inputs of size αn is connected to at least βn different outputs.

Proof: We take a series of expanders and identify the outputs $\mathcal{O}_1$ of the first one with the inputs I_2 of the second one, the outputs $\mathcal{O}_2$ of the second one with the inputs I_3 of the third one, and so on. Then there is a least integer k (independent of n) such that any set of αn inputs of I_1 is connected to at least βn different outputs of $\mathcal{O}_k$. Let c be the expansion factor. We can easily calculate k by computing the series defined by $a_0 = \alpha$ and $a_{i+1} \leftarrow a_i\,(1 + c(1 - a_i))$; then k is the smallest index i such that $a_i \geq \beta$.

Hence, to get the desired bipartite graph, we only have to connect each node v of I_1 to all nodes w of $\mathcal{O}_k$ which can be reached from v by traversing a path which uses exactly one edge from each of the k expanders. The degree of any node is clearly constant. To make the graph $d_{(\alpha,\beta)}$-regular we can add arbitrary dummy edges without destroying the expansion property. Furthermore, the expansion factor $\gamma_{(\alpha,\beta)}$ of our new expander is $\frac{\frac{\beta}{\alpha}-1}{1-\alpha}$ which we get by solving for c in $\alpha\,(1 + c(1 - \alpha)) = \beta$. ∎

We are most interested in (strong) expanders with the parameters $(n, d_{(\alpha,\beta)}, \gamma_{(\alpha,\beta)})$ where $\gamma_{(\alpha,\beta)} = \frac{1-\epsilon}{\epsilon}$ for some constant $\epsilon : 1 > \epsilon > 0$. Such expanders are used for building components of the $O(n \lg n)$ comparator and $O(\lg n)$ depth parallel sorting network of Ajtai, Komlós, and Szemerédi [2, 3].

3 Finding $O(n)$-time c-Approximate Medians for Nuts and Bolts

In this section we give the details of our algorithm for finding the c-approximate median with $O(n)$ nut-and-bolt comparisons by using ϵ-halvers.

Briefly, given a list X of $2n$ elements an ϵ-halver [2, 3, 15] approximately splits X in half with most of the small elements (at least $(1-\epsilon)n$) ending up on the right half and most of the large elements (at least $(1-\epsilon)n$) ending up on the left half. However, the ϵ-halvers must be modified so that they will always compare nuts to bolts.

There are two basic difficulties with ϵ-halving nuts and bolts that we must overcome in order to find an approximate median. We will find an approximate median from smaller and smaller lists of nuts and bolts that we get through ϵ-halving and some other operations. The first of these difficulties is that we must be able to deal with the ϵ-errors as we find an approximate median. We must not allow these errors to prevent us from finding an approximate median. Hence we must ensure that the errors diminish appropriately as our algorithm runs so that we can find an approximate median. The second difficulty we must overcome is that we must make sure that the diminishing sets of nuts and bolts, that we use to isolate an approximate median, always contain enough appropriate nuts and bolts to allow us to continue to ϵ-halve.

A comparator network has wires $w_1, w_2, \cdots, w_n, w_{n+1}, \cdots, w_{2n}$, see for example [2, 3, 15, 8, 11]. The wires $w_1, w_2, \cdots, w_n$ are *low* wires and the wires $w_{n+1}, \cdots, w_{2n}$ are *high* wires. The low wires are on the left side of the network and the high wires are on the right side of the network.

A comparator $\mathcal{C}$ between a low wire w_i and a high wire w_j puts the higher value in w_j and the lower value in w_i. A comparator network has r levels of comparators, where r is a constant. That is, at every level $\ell : r \geq \ell \geq 1$, there are n comparators among disjoint wires. Two wires are *comparable* iff one contains a nut and the other contains a bolt. Likewise, we say that two elements are *comparable* iff one is a nut and the other is a bolt. It is important to note that each level of comparators forms a (bipartite) 1-factor between the high and low wires. In a bipartite graph, a 1-factor is the same as a perfect matching.

In general, let $X[i,j] = X[i, i+1, \cdots, j]$ and we will use set-theoretic notation freely with such lists. Given a list $X[1,m]$ of m elements, we say that it is *halved* when the $\lfloor m/2 \rfloor + 1$ largest elements are in $X[\lfloor m/2 \rfloor + 1, m]$ and the $\lfloor m/2 \rfloor$ smallest elements are in $X[1, \lfloor m/2 \rfloor]$. Where an ϵ-halved list is a halved list that may contain a certain number of errors for varying sized sublists.

Definition 1 For some constant $\epsilon < 1$, let $\mathcal{S}_k$ denote the k smallest numbers in X and let $\mathcal{L}_k$ denote the k largest numbers in X. Then X is ϵ-halved iff for all $k \leq \frac{m}{2}$ we have

$$|\mathcal{S}_k \cap X[\lfloor m/2 \rfloor + 1, m]| \leq \epsilon k \quad \text{and} \quad |\mathcal{L}_k \cap X[1, \lfloor m/2 \rfloor]| \leq \epsilon k$$

An *ϵ-halver* is a comparator network that ϵ-halves its input using only $O(m)$ comparators

However, building such a comparator network in a straightforward way does not seem to give an ϵ-halver for nuts and bolts in the worst case. In particular, after the first level of comparators a standard ϵ-halver might only compare nuts with nuts and bolts with bolts for all subsequent levels of comparators.

Definition 2 For some constant $\epsilon < 1$, let $\mathcal{S}_k^N$ denote the k smallest nuts in X and let $\mathcal{S}_k^B$ denote the k smallest bolts in X. Likewise, $\mathcal{L}_k^N$ and $\mathcal{L}_k^B$ are the k largest nuts and the k largest bolts, respectively.
Then X is ϵ-halved iff for all $k \leq \frac{m}{2}$ we have

$$\left|(\mathcal{S}_k^N \cup \mathcal{S}_k^B) \cap X[\lfloor m/2 \rfloor + 1, m]\right| \leq \epsilon 2k \quad \text{and} \quad \left|(\mathcal{L}_k^N \cup \mathcal{L}_k^B) \cap X[1, \lfloor m/2 \rfloor]\right| \leq \epsilon 2k$$

A nut-and-bolt *ϵ-halver* is a comparator network using only $O(m)$ comparators that ϵ-halves its inputs of nuts and bolts. Nut-and-bolt *ϵ-halvers* are supplemented with the machinery to tell the difference between nuts and bolts and to deal with incomparable elements.

Following Definition 2, from here on all ϵ-halvers are nut-and-bolt ϵ-halvers, unless otherwise noted.

Since

$$\left|(\mathcal{S}_k^N \cup \mathcal{S}_k^B) \cap X[\lfloor m/2 \rfloor + 1, m]\right| \leq \epsilon 2k$$

we know that in the worst case

$$\left|\mathcal{S}_k^N \cap X[\lfloor m/2 \rfloor + 1, m]\right| \leq \epsilon' k$$

for some constant $\epsilon' \leq 2\epsilon$. Likewise for

$$\left|\mathcal{S}_k^B \cap X[\lfloor m/2 \rfloor + 1, m]\right| \leq \epsilon'' k$$

for some constant $\epsilon'' \leq 2\epsilon$. Hence, we let $\mathcal{S}_k$ denote the set of k smallest nuts and bolts when convenient. Naturally, the same holds for $\mathcal{L}^N$ and $\mathcal{L}^B$.

Let L_i^N and L_i^B denote the nuts and bolts immediately before comparator level i on low wires. Likewise, let H_i^N and H_i^B denote the nuts and bolts immediately before comparator level i on high wires.

If a nut matches a bolt, then either the nut or the bolt wins. In this case whichever one wins is not relevant for the correctness of our c-approximate median algorithm.

We will show how to construct non-dynamic networks shortly. A *non-dynamic* network is a comparator network that can have all of its comparator connections designated in advance before the algorithm is run. However, such a network has components for counting nuts and bolts and switching to different comparator levels depending on the numbers of high nuts or low nuts.

Given n nuts and n bolts, we want to show that comparator networks exist that ϵ-halve the nuts and bolts. First we will describe dynamically built comparator networks that exhibit graph expanding properties. Then we will give a non-dynamic nut-and-bolt ϵ-halving network.

Prior to level 1 of the comparators, for the inputs to the comparator network we put all of the nuts on the low wires $(w_1, \cdots, w_n)$ and we put all of the bolts on the high wires $(w_{n+1}, \cdots, w_{2n})$.

On comparator level 1 we just choose a random permutation π_1 on n elements. We take this permutation to describe which low wires to connect to which high wires. In the second and subsequent levels we must be careful to ensure that we only allow permutations that will describe connections between wires containing comparable elements. As a first attempt, to do this for level $i \geq 2$ we consider two random permutations: π_i^1 and π_i^2 where $\pi_i^1 : L_i^N \to H_i^B$ and $\pi_i^2 : L_i^B \to H_i^N$. These permutations tell which wires to put a comparator between, at level $i \geq 2$.

Lemma 2 For every level i of our comparator network, $n = \left|L_i^N \cup L_i^B\right| = \left|H_i^N \cup H_i^B\right|$ and $\left|L_i^N\right| = \left|H_i^B\right|$ and $\left|L_i^B\right| = \left|H_i^N\right|$.

Proof: By induction on the levels of the comparator network. ∎

Pinsker [17], Chung [9], and Pippenger [18] and others showed that expanders (and related combinatorial objects) exist using randomized methods. Here we generalize this result to be suitable for the nuts and bolts problem, while following closely the exposition given in Sarnak [21] and Lubotzky [13].

In the proof of the next theorem we show the existence of dynamically built comparator networks which are expanders that form ϵ-halvers. However, following the proof of this theorem, we show the existence of non-dynamic networks for any number of n nuts and n bolts (where n is sufficiently large).

Theorem 2 (Most dynamic random nut-and-bolt comparator networks are expanders) Let $w_1, \cdots, w_n, w_{n+1}, \cdots, w_{2n}$ be $2n$ wires in an r-level comparator network, where at each level the neighbors of the vertices $w_1, \cdots, w_n$ are chosen from $w_{n+1}, \cdots, w_{2n}$ by a random permutation which only allows wires containing comparable elements to have a comparator between them. Now, consider each wire to be a node and each comparator to be an edge in an r-degree regular bipartite graph $G = (V_1 \cup V_2, E)$ where $V_1 = \{w_1, \cdots, w_n\}$ and $V_2 = \{w_{n+1}, \cdots, w_{2n}\}$. Then with high probability G is an expander, with expansion factor $c = \frac{1}{2}$.

Proof: The set of inputs is $\mathcal{I} = V_1$ and the set of outputs is $\mathcal{O} = V_2$.

Take the r-tuple of permutations $\pi = (\pi_1, (\pi_2^1, \pi_2^2), \cdots, (\pi_r^1, \pi_r^2))$, where π_1 is chosen at random and each permutation (π_i^1, π_i^2) for $i \geq 2$ is such that each π_i^1 and π_i^2 are chosen at random depending on the contents of the wires at level i of the comparator network. In particular, for $i \geq 2$ we consider pairs of permutations. Let these two permutations be π_i^1 and π_i^2. Then there are two disjoint sets X and Y such that $X \subseteq \{1, 2, \cdots, n\}$ and $Y \subseteq \{1, 2, \cdots, n\}$ where $X \cup Y = \{1, 2, \cdots, n\}$ and $\pi_i^1 : X \to X$ and $\pi_i^2 : Y \to Y$.

This means there are $n!$ different choices for π_1 since $L_1^B = H_1^N = \emptyset$, while there are $\left|L_i^N\right|!\ \left|L_i^B\right|!$ different choices for (π_i^1, π_i^2) for $i : r \geq i \geq 2$, see Lemma 2. Considering that we must choose how large π_1 and π_2 are, we really have

$$\binom{n}{\left|L_i^N\right|} \left|L_i^N\right|!\ \left|L_i^B\right|!$$

choices for level i.

Now, we will bound the number of r-tuples π where there is a subset $A \subseteq \mathcal{I}$ such that $|A| \leq n/2$ and at the same time $\pi_i(A) \subseteq C$ for all i where $C \subseteq \mathcal{O}$ and $|C| \leq \frac{3}{2}|A|$. Any r-tuple of permutations that allows any such A and C is a *bad* r-tuple. Note, that we are going to show that almost all such graphs are expanders with expansion factor $c = \frac{1}{2}$.

Let $|A| = s$ and $|C| = t$, with $t \leq \frac{3}{2}s$. To count the number of bad r-tuples consider the sets of wires containing comparable items at the i-th round. If A is the set of inputs that are only mapped to C through all r rounds, where $|C| \leq \frac{3}{2}|A|$, we let A_i^N denote the subset of A that contains all nuts of A in round i and A_i^B denote the subset of A that contains all bolts of A in round i. Likewise, let C_i^N denote the subset of C that contains all nuts of C in round i and C_i^B denote the subset of C that contains all bolts of C in round i.

That is,

$$\begin{array}{cc} A_i \subset L_i^N \cup L_i^B & C_i \subset H_i^N \cup H_i^B \\ A_i^N \subseteq L_i^N & A_i^B \subseteq L_i^B \\ C_i^N \subseteq H_i^N & C_i^B \subseteq H_i^B \end{array}$$

and of course the first time we run any expander we have $A_1 = A_1^N \subseteq L_1^N$ and $C_1 = C_1^B \subseteq H_1^B$, since $L_1^B = H_1^N = \emptyset$.

Now, let $h(a^N, a^B, c^N, c^B)$ denote the number of "bad permutations" for a random nuts-and-bolts bipartite graph. From here on we let x^Y denote the cardinality of the set X^Y. Further, when convenient we drop subscripts so that we denote X_i^Y as X^Y when there will be no ambiguity.

Then

$$h(a^N, a^B, c^N, c^B) \quad = \quad \prod_{i=1}^{r} \Big((|L_i^N| - a_i^N)! \; (|L_i^B| - a_i^B)! \Big) \prod_{i=1}^{r} h_1(a^N, c^B, i) h_2(a^B, c^N, i)$$

where $a^N + a^B = s$ and $c^N + c^B = t$ and

$$\begin{aligned} h_1(a^N, c^B, i) &= c_i^B(c_i^B - 1) \cdots (c_i^B - a_i^N + 1) \binom{s}{a_i^N} \\ h_2(a^B, c^N, i) &= c_i^N(c_i^N - 1) \cdots (c_i^N - a_i^B + 1) \binom{t}{c_i^N} \end{aligned}$$

where h_1 denotes the number of choices that the nuts in A have on compatible elements in C. Likewise for h_2. (We are assuming without loss that $c_i^B \geq a_i^N$ and $c_i^N \geq a_i^B$ since all permutations must map from A into C.)

Let $N(s,t)$ denote the number of bad permutations for a comparator network, so we have

$$N(s,t) \quad = \quad \sum_{s \leq \frac{n}{2}} \left(\sum_{s \leq t \leq \frac{3}{2}s} \binom{n}{s} \binom{n}{t} h(a^N, a^B, c^N, c^B) \right).$$

Let D denote the number of possible different networks, so we have

$$D \quad = \quad \prod_{i=1}^{r} \left(\binom{n}{|L_i^N|} |L_i^N|! \;\; |L_i^B|! \right).$$

Note that since $|L_i^B| = n - |L_i^N|$ we really have

$$\binom{n}{|L_i^N|} |L_i^N|! \; (n - |L_i^N|)! \quad = \quad n!.$$

Hence, $D = (n!)^r$.

So, a bound on the number of bad r-tuples divided by the exact number of possible r-tuples is:

$$\frac{N(s,t)}{D} = \frac{\displaystyle\sum_{s\le\frac{n}{2}}\left(\sum_{s\le t\le\frac{3}{2}s}\binom{n}{s}\binom{n}{t}h(a^N,a^B,b^N,b^B)\right)}{(n!)^r}.$$

Following Sarnak [21] and Lubotzky [13], we consider two cases, the first where $s \le \frac{n}{3}$ and then the case where $n/3 < s \le n/2$.

So, now we show that $N(s,t) \ge N(s+1,t)$ for $s \le \frac{n}{3}$.

As in both cases we start with,

$$\binom{n}{s}\binom{n}{t} = \frac{n!\ n!}{s!\ (n-s)!\ t!\ (n-t)!}.$$

Now considering that

$$h_1(a^N,c^B,i) = c_i^B(c_i^B-1)\cdots(c_i^B-a_i^N+1)\binom{s}{a_i^N} \tag{1}$$

hence

$$\prod_{i=1}^r h_1(a^N,c^B,i) = \prod_{i=1}^r\left(\frac{c_i^B!}{a_i^N!}\right)\left(\frac{s!}{a_i^N!\ (s-a_i^N)!}\right). \tag{2}$$

Likewise for h_2 we have

$$\prod_{i=1}^r h_2(a^B,c^N,i) = \prod_{i=1}^r\left(\frac{c_i^N!}{a_i^B!}\right)\left(\frac{t!}{c_i^N!\ (t-c_i^N)!}\right) \tag{3}$$

and we recall that $s = a_i^N + a_i^B$. Therefore, with the other factors let

$$M(s,t) = \binom{n}{s}\binom{n}{t}\prod_{i=1}^r\Big(h_1(a^N,c^B,i)h_2(a^B,c^N,i)\Big)\prod_{i=1}^r\Big((|L_i^N|-a_i^N)!\ \ (|L_i^B|-a_i^B)!\Big). \tag{4}$$

We want to show that $M(s,t)$ has its maximum value at $M(1,t)$ which would mean $N(s,t) \ge N(s+1,t)$ for $s \ge 1$ which gives us a sufficient upper bound. We do this by showing that $M(s,t)$ maximizes when both a_i^N and a_i^B are small. If both a_i^N and a_i^B are small, then s is also small, hence $N(s,t) \ge N(s+1,t)$ will hold. Therefore, we can bound $N(s,t)$ and complete the proof.

In order to maximize $M(s,t)$ we first note in Equation (3) the c_i^N-s cancel out just as $(t-c_i^N) = c_i^B$ since $t = c_i^N + c_i^B$, so the $(t-c_i^N)!$ term cancels with the $c_i^B!$ term too. Therefore, we have to consider the function,

$$f(s,t,c_i^B,a_i^N,a_i^B) = \left(\frac{t!}{a_i^B!}\right)\binom{s}{a_i^N}\frac{(|L_i^N|-a_i^N)!\ (|L_i^B|-a_i^B)!}{a_i^N!}$$

and via straightforward manipulation we can see that for $s \ge 1$ the function f maximizes when $|L_i^N| = n$ and $a_i^N = s = 1$. Furthermore,

$$f(s,t,c_i^B,a_i^N,a_i^B) \quad \leq \quad (n-1)!$$

for $s : \frac{n}{3} \geq s \geq 1$.

We must consider this together with the

$$g(n,s,t) \quad = \quad \binom{n}{s}\binom{n}{t}$$

terms. The function $g(n,s,t)$ is maximum when $s = \frac{n}{3}$ and $t = \frac{n}{2}$, which using Stirling's approximation shows that

$$f(s,t,c_i^B,a_i^N,a_i^B) \quad > \quad (n/2)!$$

holds for each level i of the network.

From this we conclude that $N(s,t) \geq N(s+1,t)$ by comparing the minimal value of f with the maximal values of the denominator of g.

Now, we finally claim,

$$\lim_{n\to\infty} \frac{N(1,t)}{(n!)^r} \quad \to \quad 0. \tag{5}$$

Establishing this claim shows that for $s \leq n/3$ most such dynamically-built and randomly chosen graphs must be good expanders, since the ratio of the number of bad permutations over the number of permutations goes to zero as n grows large.

The case for $\frac{n}{3} < s \leq \frac{n}{2}$ follows similarly, see also Sarnak [21]. (It is not even necessary to write it out in full, since it is well known that expanders with $|A| \leq \frac{n}{3}$ can be iterated into expanders with $|A| \leq \frac{n}{2}$.)

We establish the above claim in Equation (5) as follows. First, we claim

$$N(1,t) \quad \leq \quad n^2 \prod_{i=1}^{r} \Big((|L_i^N| - a_i^N)! \ \ (|L_i^B| - a_i^B)! \Big)$$

Since $s = 1$, we know that both $|L_i^N| - a_i^N < n$ and $|L_i^B| - a_i^B < n$ because $|L_i^N| + |L_i^B| = n$. Hence

$$\lim_{n\to\infty} \frac{N(1,t)}{(n!)^r} \quad \to \quad 0$$

for each $i : r \geq i \geq 1$ where r is a constant larger than 4. This implies Equation 5, completing the proof. ∎

Theorem 2 shows the existence of such dynamically built nut-and-bolt expanders with expansion factor $c = \frac{1}{2}$. However, we will show that for every input of n nuts and n bolts there are fixed networks that are nut-and-bolt ϵ-halvers.

Definition 3 A set U of nuts or bolts *illicitly supports* a set V of elements if the elements in V only make comparisons with elements of U and because of these comparisons, the elements in V are placed on the wrong side of the comparator network.

It is interesting to note that ϵ-halving a list of n elements, there can be at most ϵn illicitly supported elements. Further, in another ϵ-halving of one side of this list, these ϵn illicitly supported elements, can illicitly support at most $\left(\frac{\epsilon}{1-\epsilon}\right) \epsilon n$ elements.

Theorem 3 Given n nuts and n bolts, where n is sufficiently large, there exist non-dynamic comparator networks that are ϵ-halvers. Moreover, these nut-and-bolt ϵ-halvers are of constant depth and they use a total of $O(n)$ comparators.

Proof: We will show that our ϵ-halving comparator network is non-dynamic and consists of 1-factors between the low and high wires.

For each comparator level $j \geq 2$ the first 1-factor of each pair of 1-factors describes comparators between the nuts L_j^N and bolts H_j^B. The second 1-factor of each pair of 1-factors describes comparators between the bolts L_j^B and nuts H_j^N. After a comparator level j, we can count the number of nuts and bolts in L_j^N and L_j^B and we send nuts L_j^N and bolts H_j^B down opposite sides of a suitable 1-factor. In this case, in the first 1-factor of the pair of 1-factors at level j, we send the list of nuts in L_j^N down the low wires (right side) of this 1-factor and the list of bolts in H_j^B down the high wires (left side). We choose the 1-factor for L_j^N and H_j^B as the 1-factor-based comparator that the number $|L_j^N|$ is closest to but not greater than in size. By our construction the second 1-factor will be large enough to suit all of the comparisons between the bolts L_j^B and the nuts H_j^N. Send L_j^B to the right and H_j^N to the left of the second 1-factor. The 1-factors are a little oversized, so that we don't have to throw any nuts or bolts away. But, we do duplicate some nuts and bolts to "fill out" to the size of the 1-factors at each level. We dispose of these extra nuts and bolts and their associated wires, after the last level of the comparator. Then we re-group $L_{j+1}^N, L_{j+1}^B, H_{j+1}^N$ and H_{j+1}^B and continue.

In particular, for each $i : \left\lceil \frac{1}{\epsilon} \right\rceil \geq i \geq 1$ there is a pair of 1-factors among the following number of elements: $\lceil (1-i\epsilon)n \rceil$ and $\lceil (i2\epsilon)n \rceil$. That is, there are two bijections, one among $\lceil (1-i\epsilon)n \rceil$ elements and the other among $\lceil (i2\epsilon)n \rceil$ elements.

Let r be the number of levels in our comparator network. Now, we will have at most $r2\epsilon$ extra copies of nuts and bolts coming out of the last level of comparators. Furthermore, in the worst case these extra $r2\epsilon$ nuts and bolts can "illicitly support" at most

$$\left(\frac{1}{\beta}\right) 2r\epsilon \;=\; \frac{2r\epsilon^2}{1-\epsilon}$$

other bolts and nuts on the incorrect side of the comparator. For appropriately chosen ϵ, this has no effect on the correctness of our algorithm because this is just an adjustment to the value of ϵ.

Finally, we note that Theorem 2 guarantees the existence of pre-chosen graphs which are comparator networks with the required expansion properties. ∎

We could find such expanders explicitly *without* any nut-and-bolt comparisons. We can certainly find them in exponential time by enumerating all r-level comparator networks and considering all appropriate permutations on each level and then by checking all appropriate subsets for the expansion property, etc. Also, see the citations in the introduction, and in particular Pippenger [19].

Finally, we note that the proof of Theorem 2 does not show that *strong* nut-and-bolt expanders exist, but just that expanders exist. By Lemma 1 we know that strong expanders exist too. (To apply Lemma 1, we have to show that all subsets of both the inputs and outputs of size up to $n/2$ expand, but this is straightforward by symmetry from the proof above. See also [4, Lemma 4.1].)

Following Corollary 1, using the results just discussed we can construct expanders with parameters $(n, d_{(\alpha,\beta)}, \gamma_{(\alpha,\beta)})$ where $\gamma_{(\alpha,\beta)} = \frac{1-\epsilon}{\epsilon}$ for some constant $\epsilon : 1 > \epsilon > 0$. Furthermore, we know that $d_{(\alpha,\beta)}$ is some constant based only on α and β.

The next observation is a minor variation of a classical result [2, 3].

Observation 4 (See [2, 3, 15]) Let $i : 1 \leq i \leq n$ and $j : n+1 \leq j \leq 2n$. If w_i and w_j have a comparator between them at level K, then $\mathit{Output}(w_i) \leq \mathit{Output}(w_j)$ on every subsequent level $K+1, \cdots$, even if in subsequent levels the contents of w_i and w_j are not comparable.

Proof: First, we only compare between low wires and high wires which contain comparable elements. That is, two wires w_i and w_j can have a comparator between them iff $1 \leq i \leq n$ and $n+1 \leq j \leq 2n$ and both wires are carrying comparable elements.

Suppose w_i is such that $i \leq n$ and w_j is such that $j \geq n+1$. One round after there is a comparator between two wires w_i and w_j. These wires may no longer contain comparable elements. However, even in this case, the only way to exchange the contents of w_i is to replace it with a smaller (or matching) element, and similarly the only way to replace the contents of w_j is to replace it with a larger (or matching) element. ∎

We begin by nut-and-bolt ϵ-halving a list $X[1, 2n]$, then continuing on its left half $X[1, n]$, and then nut-and-bolt ϵ-halving $X[1, n]$, then continuing on its right half $X[n/2, n]$, etc. That is, first we ϵ-halve our current elements, in even iterations we choose to continue on the right half and during odd iterations we choose to continue on the left half. Hence, in each iteration we halve the number of elements that we consider. We will also show how to build routines to get the extraneous nuts and bolts, if there are any.

The nuts and bolts we are considering in the i-th iteration are in the list X_i. The position an element is in X_i indicates which wire it is on. So X_0 is the given list of n nuts and n bolts where all of the nuts are on the low wires and all of the bolts are on the high wires. Repeatedly using ϵ-halvers on geometrically smaller sets of the most recent set of halved elements we get the sequence of nuts and bolts: $X_0, X_1, \cdots, X_i$ where $i : \lceil \lg n \rceil \geq i$ and $|X_{i+1}| \leq \lceil |X_i| / k \rceil$, where $k > \frac{2}{1+K\epsilon}$, for K a small constant such that $K\epsilon < 1$ and K will be defined later. This is because we will continually add some nuts and bolts back to the ϵ-halved lists. For ease of exposition and without loss of generality we will take $k = 2$, though it is sufficient to take it as $k = 1.5$.

If we ϵ-halve $X[1, 2n]$, then for s to be in $X[1, n]$ with no match in $X[1, n]$, either s or its matching element t must have been put on the wrong side of $X[1, 2n]$ after it was ϵ-halved. Hence, for an element to have no matching element on the same side, must be the result of one of the bounded number of "errors" the ϵ-halving allows.

We will write

$$\mathcal{S}_k = \mathcal{S}_k^N \cup \mathcal{S}_k^B \quad \text{and} \quad \mathcal{L}_k = \mathcal{L}_k^N \cup \mathcal{L}_k^B$$

when it is convenient and it does not introduce any ambiguity.

Suppose we ϵ-halve $X[1, 2n]$. Then the errors, say $E_{\mathcal{L}}$, in $X[1, n]$ are the elements that are too large, that is elements from $\mathcal{L}_{n/2+1}$. We will show that $|E_{\mathcal{L}}|$ is very small. The elements $E_{\mathcal{L}}$ may not have matches in $X[1, n]$, however, they will be comparable with a lot of the elements in $X[1, n]$. When $X[1, n]$ is ϵ-halved, then "most" of the elements in $E_{\mathcal{L}}$ will end up in $X[n/2, n]$. Of course, after we ϵ-halve $X[n/2, n]$, then we will continue on $X[n/2, 3n/4]$. This forces the elements in $E_{\mathcal{L}} \cap X[n/2, 3n/4]$ to diminish substantially more in number.

Likewise, the errors in $X[n/2, n]$ will be small elements from $\mathcal{S}_{n/2}$ which are "too small" for $X[n/2, n]$. Call these too small elements $E_{\mathcal{S}}$. Note that $|E_{\mathcal{S}}|$ is very small. Furthermore, by the time we ϵ-halve $X[n/2, 3n/4]$ and consider the list $X[5n/8, 3n/4]$, we have diminished the number of elements in $E_{\mathcal{S}} \cap X[5n/8, 3n/4]$ substantially more.

Definition 4 An *extraneous* element is an element that is not in the present list of expected sizes if each ϵ-halve was an exact halving, that is $\epsilon = 0$.

This definition holds for the case where the extraneous elements are from outside of the present range.

When we have

$$\left|\mathcal{S}^N_{n/2} \cap X\right| \;\; \geq \;\; (1-\delta)n/2$$

for $\frac{1}{8} \geq \delta$, this means "most" of the small nuts are in the list X. Considering all of the sizes of nuts and bolts; we have the following

$$\begin{array}{ccc} \left|\mathcal{S}^N_{n/2}\right| = n/2 & \text{and} & \left|\mathcal{S}^B_{n/2}\right| = n/2 \\ \left|\mathcal{L}^N_{n/2+1}\right| = n/2 & \text{and} & \left|\mathcal{L}^B_{n/2+1}\right| = n/2 \end{array}$$

and the set

$$\mathcal{S}^N_{n/2} \cup \mathcal{S}^B_{n/2} \cup \mathcal{L}^N_{n/2+1} \cup \mathcal{L}^B_{n/2+1}$$

is all of the given $2n$ nuts and bolts in $X_0[1, 2n]$ for the nuts and bolts problem.

The next lemma, when $\delta = 0$, is just the same as the standard ϵ-halving theorem adapted to nuts and bolts, see [2, 3, 15].

Lemma 3 Let ϵ be some constant such that $1 > \epsilon > 0$ and let δ be any constant such that $\frac{1}{8} \geq \delta \geq 0$. Given a list of m nuts and m bolts in a list X such that $(1-\delta)m$ of the nuts have matching bolts in X, then we can build a nut-and-bolt ϵ-halver of $O(m)$ comparators for ϵ-halving the nuts and bolts.

Proof: Let $\mathcal{S}^N_k$ denote the smallest nuts of size at most k expected to be in X. Let $\overline{\mathcal{S}^N_k}$ denote the nuts not larger than the k-th smallest nut in $\mathcal{S}^N_k$. Since there are at most δm nuts and bolts that are not among any of the remaining $(1-\delta)m$ nuts and bolts in $\mathcal{S}^N_{m/2} \cup \mathcal{S}^B_{m/2}$, without loss of generality we assume that $\left|\overline{\mathcal{S}^N_k}\right| = k + \delta m/2$ and likewise $\left|\overline{\mathcal{S}^B_k}\right| = k + \delta m/2$. We can assume this since the $2\delta m$ extraneous elements can't be between $\mathcal{S}_{m/2}$ and $\mathcal{L}_{m/2+1}$ in size (since there are no sizes between $\mathcal{S}_{m/2}$ and $\mathcal{L}_{m/2+1}$ in X).

Similarly, the definitions for $\overline{\mathcal{L}^N_k}$ and $\overline{\mathcal{L}^B_k}$ follow in the expected way and we assume without loss that $\left|\overline{\mathcal{L}^N_k}\right| = \left|\overline{\mathcal{L}^B_k}\right| = k + \delta m/2$.

By Theorem 3, nut and bolt ϵ-halvers exist, and now we show that they can tolerate some nuts and bolts that have no matches.

Let $W \subseteq \{w_{m+1}, \cdots, w_{2m}\}$ be the set of high wires that contains elements of $\overline{\mathcal{S}^N_k} \cup \overline{\mathcal{S}^B_k}$ after the last level of comparators. Let $\overline{W} \subseteq \{w_1, \cdots, w_m\}$ be the set of low wires that shared a comparator with a wire in W at some level.

Claim 1: Each wire in W carries an element of $\overline{\mathcal{S}^N_k} \cup \overline{\mathcal{S}^B_k}$ at every level. Considering Observation 4, a proof by contradiction is immediate.

Claim 2: Each wire in $\overline{W}$ carries an element of $\overline{\mathcal{S}^B_k} \cup \overline{\mathcal{S}^N_k}$ after the last level of the comparator network. Considering Observation 4, a proof by contradiction is immediate.

Now, Claims 1 and 2 give the following bound on the number of errors left by the ϵ-halver.

Main Claim: $|W| \leq \epsilon(2k + \delta m)$.

We can show this in the standard way as follows. For the sake of a contradiction, suppose $|W| > \epsilon(2k + \delta m)$. By the expansion properties of the graph that was constructed we have:

$$\left|\overline{W}\right| \geq \beta|W| > \beta\epsilon(2k + \delta m).$$

But, since $\overline{W} \cap W = \emptyset$, we know that

$$\left|\overline{W} \cup W\right| > (\beta + 1)\epsilon(2k + \delta m).$$

This means $\left|\overline{\mathcal{S}_k^N} \cup \overline{\mathcal{S}_k^B}\right| > (\beta+1)\epsilon(2k+\delta m)$ which can't be since we chose $\beta = \frac{1-\epsilon}{\epsilon}$ and $\left|\overline{\mathcal{S}_k^N} \cup \overline{\mathcal{S}_k^B}\right| = 2k + \delta m$.

The symmetric case with $\overline{\mathcal{L}_k^N} \cup \overline{\mathcal{L}_k^B}$ also holds. Finally, this is all done with a nuts-and-bolts ϵ-halver so that it costs $O(m)$ comparisons, completing the proof. ▮

Lemma 3 shows that the fraction of extraneous nuts and bolts don't add substantially to the the bounded number of errors for ϵ-halving. Naturally, we are assuming that δm is small. At first, for all n nuts and n bolts, each nut has a matching bolt. Our algorithm will take geometrically smaller lists of nuts and bolts and nut-and-bolt ϵ-halve them. The **Main Claim** in the proof of Lemma 3 gives bounds on the numbers of "errors" among elements from a nut-and-bolt ϵ-halver. We must take special care to ensure that enough comparable nuts and bolts remain in our list. We will show how to do this shortly.

Let $E_{\mathcal{L}}$ denote the bounded number of errors from larger elements $\mathcal{L}$, where the elements $E_{\mathcal{L}}$ are erroneously in the lower half of the ϵ-halved list. Similarly, let $E_{\mathcal{S}}$ denote the bounded number of errors from smaller elements $\mathcal{S}$, where the elements $E_{\mathcal{S}}$ are erroneously in the higher half of the ϵ-halved list.

The algorithm Get-c-Approximate-Median in Figure 1 uses two key functions Back-Track and Find_Misplaced_Elements that will be defined shortly. They are applied in each iteration of Get-c-Approximate-Median and basically they get back as many "useful" elements as possible that are misplaced by the ϵ-halving. (We will discuss this in more detail shortly.) They allow us to continually ϵ-halve the main list X.

To complete the proof of the existence of the $O(n \lg n)$ nut-and-bolt comparisons algorithm we will show that the algorithm Get-c-Approximate-Median maintains the following invariant.

Invariant 1 For all iterations $i : \lceil \lg n \rceil \geq i \geq 0$, the list X_i contains at least one c-approximate median.

Showing that this invariant holds will take the lion's share of the rest of this paper. To do this, we will show how to deal with the recurring ϵ-errors due to ϵ-halving and we must also show how to keep our on-going list containing medians so that it can continually be ϵ-halved.

The algorithm Get-c-Approximate-Median never runs more than $O(\lg n)$ iterations by our choices of ℓ and r, so each time we ϵ-halve $X[\ell, r]$, we expend half of the work of the previous time. We briefly mention that we will always have comparable elements in X_i as long as $|X_i| \geq C$. We may choose the size of C depending on the trade-off between the size of C and the overhead associated with using the expanders.

We introduce $2(K\epsilon)n/2^i$ elements in the i-th step back into X_i, where $K\epsilon < 1$. Furthermore, always adding these elements in does not change the asymptotic complexity of our algorithm.

The intervals of X_i that we will iterate on are denoted as $X[\ell(i), r(i)]$ for the i-th iteration where $\ell(i)$ is the left boundary and $r(i)$ is the right boundary. We write out a few terms of $[\ell(i),\ r(i)]$ in the following table. (We will later see that the values for ℓ and r are off by an $K\epsilon$ factor, but this makes no difference asymptotically.)

```
Get-$c$-Approximate-Median($X$)
    $n \leftarrow |X_0|/2$
    $\ell \leftarrow 1$;    $r \leftarrow 2n$
    $i \leftarrow 0$
    **While** $|X_i| \geq C$ **do**    (* $C$ is a constant *)
        $Y_i \leftarrow$ nut-and-bolt-$\epsilon$-halve($X_i$)
        $B \leftarrow$ Back-Track($Y_i, i, Z$)
        $Z \leftarrow$ Find_Misplaced_Elements($Y_i, i, n/2^i$)
        **if** $i$ is odd **then**
            $r \leftarrow (\ell + r)/2$    (* Right boundary of $Y^i$ *)
        **else**
            $\ell \leftarrow (\ell + r)/2$    (* Left boundary of $Y^i$ *)
        **endif**
        $i \leftarrow i + 1$
        $X_i \leftarrow Y_{i-1}[\ell,\ r] \cup Z \cup B$
    **od**
Return $X_i$
```

Figure 1: Selecting a c-approximate median with $O(n)$ work

$[\ell(0), r(0)]$,	$[\ell(1), r(1)]$	$[\ell(2), r(2)]$	$[\ell(3), r(3)]$	$[\ell(4), r(4)]$	$[\ell(5), r(5)]$
$[1, 2n]$	$[1, n]$	$[n/2, n]$	$[n/2, 3n/4]$	$[5n/8, 3n/4]$	$[5n/8, 11n/16]$

Lemma 4 Let $i > j$ and suppose $i - j = 2r$ for some $r \geq 1$ in the algorithm Get-c-Approximate-Median, where the sets $E_{\mathcal{L}_{n/2+1}}$ and $E_{\mathcal{S}_{n/4}}$ were given at step j. Assuming that nut-and-bolt ϵ-halving can be maintained throughout each iteration of Get-c-Approximate-Median then we have

$$\left| E_{\mathcal{L}_{n/2+1}} \cap X_i \right| \leq 2\epsilon^{(i-j)} n$$

and

$$\left| E_{\mathcal{S}_{n/4}} \cap X_i \right| \leq 2\epsilon^{(i-j+1)} n.$$

Proof: We only show the first case, the second case follows almost identically. As the lemma statement says, we assume that the list is always ϵ-halvable. This is by induction on the size of the difference $i - j$.

Basis: Take the case where $i - j = 2$. Here we have the following.

First, ϵ-halve the list $X_0[1, 2n]$ and then continue on $X_1[1, n]$. We know

$$\left| E_{\mathcal{L}_{n/2+1}} \cap X_1[1, n] \right| \leq 2\epsilon n$$

by the ϵ-halver properties and by Lemma 3. Now, ϵ-halve $X_1[1, n]$ and then continue on $X_2[n/2, n]$. Start by ϵ-halving $X_2[n/2, n]$ and then consider $X_3[n/2, 3n/4]$. We know

$$\left| E_{\mathcal{L}_{n/2+1}} \cap X_3[n/2, 3n/4] \right| \leq 2\epsilon^2 n$$

by the ϵ-halver properties and by Lemma 3. This completes the base case.

Inductive Hypothesis: Take some k such that $k \geq r \geq 1$. Suppose the statement of this lemma holds for all $i > j$ such that $i - j = 2r$ and $r \geq 1$.

Inductive Step: Consider the case when $i - j = 2r = 2k - 2$ so $r = k - 1$ and take the interval $X_i[b, t]$ that we are considering at this step. By the inductive hypothesis we know that,

$$\begin{aligned} \left| E_{\mathcal{L}_{n/2+1}} \cap X_i \right| &\leq 2\epsilon^{(i-j)} n \\ &\leq 2\epsilon^{(2k-2)} n. \end{aligned}$$

Now let $t' \leftarrow \frac{b+t}{2}$ and then ϵ-halve $X_i[b, t]$ and continue on $X_{i+1}[b, t']$. We know

$$\left| E_{\mathcal{L}_{n/2+1}} \cap X_{i+1}[b, t'] \right| \leq 2\epsilon \left| E_{\mathcal{L}_{n/2+1}} \right|$$

by the ϵ-halver properties and Lemma 3. Let $b' \leftarrow \frac{b+t'}{2}$ and then ϵ-halve $X_{i+1}[b', t]$ and continue on $X_{i+2}[b', t']$. Let $t'' \leftarrow \frac{b'+t'}{2}$ and then ϵ-halve $X_{i+2}[b', t']$ and consider $X_{i+3}[b', t'']$. We know

$$\begin{aligned} \left| E_{\mathcal{L}_{n/2+1}} \cap X_{i+3}[b', t''] \right| &\leq 2\epsilon^2 \left| E_{\mathcal{L}_{n/2+1}} \right| \\ &\leq 2\epsilon^{2k-2+2} n \\ &\leq 2\epsilon^{2k} n. \end{aligned}$$

This completes the proof. ∎

One view of the significance of this last lemma is that the sets $\mathcal{S}^N_{n/4} \cup \mathcal{S}^B_{n/4}$ and $\mathcal{L}^N_{n/2+1} \cup \mathcal{L}^B_{n/2+1}$ have exponentially diminishing numbers of elements in the progressively smaller X_i-s.

Lemma 4 generalizes in a straightforward way. Although, it is enough to notice that the set $X_0 - \{\mathcal{S}_{n/4} \cup \mathcal{L}_{n/2+1}\}$ contains only c-approximate medians.

It remains to be established that the ϵ-halving properties can be maintained throughout the iterations so Lemma 4 can be applied. To maintain ϵ-halving, there are several things that must be considered. First of all, there must be enough comparable elements. That is, if we are left with a set of only nuts, then we can't continue ϵ-halving. On the other hand, the elements in the remaining list must have sizes that are "intermixed enough." For example, suppose that all of the nuts and bolts remaining are such that the nuts are all smaller than the smallest bolt. Then, in the worst case we cannot repeatedly ϵ-halve for long as our algorithm specifies while maintaining Invariant 1. Clearly, if "most" the nuts in the appropriate size range have matching bolts in each set X_i, then both of these problems are overcome for an appropriate definition of "most." We show how to ensure this with $O(n)$ nut-and-bolt comparisons.

Now we give a way to retrieve extraneous elements to allow ϵ-halving to continue throughout Get-c-Approximate-Median. To this end, we always maintain the invariant that there are at least $(1 - 8\epsilon)n/2^i$ nuts with matching bolts in the list X_i. This comes at a cost of having a "few" extra elements added back into the list X_i at each step i. Most of these extra elements are too large or too small and hence will be automatically eliminated as the algorithm continues.

Given n nuts with n matching bolts in $X_0[1, 2n]$, then after ϵ-halving them, if $2\epsilon n$ small nuts and small bolts that belong to $\mathcal{S}_{n/2}$ (so they should be in $X_1[1, n]$) are in $X_1[n + 1, 2n]$, then they must have always been compared to smaller elements in $X_1[1, n]$ at each level of the ϵ-halving comparator network. The proof of this observation is a direct result of ϵ-halving.

Definition 5 If $X_i[s,t]$ is ϵ-halved and Get-c-Approximate-Median continues on $X_{i+1}[s,(s+t)/2]$, then $X^F_{i+1}[(s+t)/2+1,t]$ is the left *fringe*. (We will use the super-script F to denote fringes.) Right fringes are defined similarly, see Figure 2.

Figure 2 shows the first left and right fringes. Later as Get-c-Approximate-Median iterates new fringes are created in the obvious manner. The *outer* right and *outer* left fringes are where illicitly supported elements will be found.

Figure 2: The most recent left and right *fringes*

We say extraneous nuts-and-bolts are *active* if they belong in none of the fringes so far. That is, active nuts-and-bolts belong to the present section of X that is going to be ϵ-halved next by Get-c-Approximate-Median. In essence, back-tracking makes the active nuts-and-bolts continually decrease geometrically in number as Get-c-Approximate-Median iterates. We may call the list X_i the active list since we will see that it contains most of the active elements.

The two algorithms Find_Misplaced_Elements and Back-Track both work on fringes. Find_Misplaced_Elements starts on each "new" fringe and ϵ-halves it a constant number of times "towards" the active part of X_i. Back-Track retrieves as many active elements that are left in the outer ("old" and "new") fringes as Find_Misplaced_Elements is run, see Figure 2.

Given a list of nuts and bolts that have been ϵ-halved we back-track in a fringe as follows.

Take the list $X_0[1,2n]$ of n nuts and n bolts and say that $X_0[1,2n]$ was just ϵ-halved into X_1. There were at most $2\epsilon n$ errors introduced into $X_1[n+1,2n]$ by the ϵ-halving of $X_0[1,2n]$. Since $X_1[n+1,2n]$ is now a fringe, we will write it as $X^F_1[n+1,2n]$ from now on (as well as other levels of the fringes).

Now, we run Find_Misplaced_Elements and it ϵ-halves the fringe $X^F_1[n+1,2n]$ into $X^F_2[n+1,3n/2]$ and $X^F_2[3n/2+1,2n]$. Altogether the $2\epsilon n$ errors in $X^F_1[n+1,2n]$ can illicitly support $2\left(\frac{\epsilon^2}{1-\epsilon}\right)n$ or fewer elements in $X_2[3n/2+1,\ 2n]$. This is because

$$\left(\frac{2}{\beta}\right)\epsilon n \quad = \quad 2\left(\frac{\epsilon^2}{1-\epsilon}\right)n.$$

If we knew the $2\epsilon n - 2\left(\frac{\epsilon^2}{1-\epsilon}\right)n$ or fewer errors in $X^F_2[n+1,3n/2]$, then we could find all of the fewer than

$$2\left(\frac{\epsilon^2}{1-\epsilon} - \frac{\epsilon^3}{(1-\epsilon)^2}\right)n$$

elements in $X_2^F[3n/2, 2n]$ that were *only* compared to these $2\epsilon n - 2\left(\frac{\epsilon^2}{1-\epsilon}\right) n$ or fewer elements when $X_1^F[n+1, 2n]$ was ϵ-halved. These $2\left(\frac{\epsilon^2}{1-\epsilon} - \frac{\epsilon^3}{(1-\epsilon)^2}\right) n$ elements or fewer are *candidates* for belonging to $X_1[1, n]$ and candidates for being illicitly supported (directly or indirectly) by the errors from ϵ-halving $X_0[1, 2n]$. The process of finding such candidates is called *back-tracking* and it requires *no* nut-and-bolt comparisons.

Back-tracking still can be done after many ϵ-halving steps have occurred. (The ϵ-halving of the fringes is done by Find_Misplaced_Elements.) For example, suppose we continue to ϵ-halve $X_2^F[n+1,\ 3n/2]$ and consider the two lists $X_3^F[n+1,\ 5n/4]$ and $X_3^F[5n/4+1,\ 3n/2]$. Without loss of generality we assume there were $2\left(\frac{\epsilon^2}{1-\epsilon}\right) n$ or fewer illicitly supported elements in $X_2^F[3n/2+1,\ 2n]$. This leaves at most

$$\left(\frac{\epsilon}{1-\epsilon}\right)\left(2\epsilon n - 2\left(\frac{\epsilon^2}{1-\epsilon}\right) n\right)$$

candidates for illicitly supported elements in $X[5n/4+1, 3n/2]$. Of course, if we know which elements make up the

$$2\epsilon n - \left(4\left(\frac{\epsilon^2}{1-\epsilon}\right) - 2\left(\frac{\epsilon^3}{(1-\epsilon)^2}\right)\right) n$$

errors remaining in $X_3^F[n+1, 5n/4]$, then we can back-track in $X_3^F[5n/4+1, 3n/2]$ finding the up to $2\left(\frac{\epsilon^2}{1-\epsilon}\right) n$ illicitly supported candidates. We do this by just seeing which subset of the outer-fringes are supported completely by either apparently active elements or in this example supported completely by known extraneous elements. Taking these

$$2\left(\frac{\epsilon^2}{1-\epsilon} - \frac{\epsilon^3}{(1-\epsilon)^2}\right) n$$

candidates and the previous

$$2\epsilon n - \left(4\left(\frac{\epsilon^2}{1-\epsilon}\right) - 2\left(\frac{\epsilon^3}{(1-\epsilon)^2}\right)\right) n$$

errors together, give a total of

$$2\epsilon n - 2\left(\frac{\epsilon^2}{1-\epsilon}\right) n$$

candidates we know in $X_2^F[n+1, 3n/2]$. With these we can back-track *again* finding the $2\left(\frac{\epsilon^2}{1-\epsilon} n\right)$ or fewer error candidates in $X_2^F[3n/2, 2n]$.

Here we point out that in the ϵ-halving of a fringe we successively have at most

$$2n\left(\epsilon - \left(\frac{\epsilon^2}{1-\epsilon}\right)\right)$$

errors, then in the next iteration we have at most

$$2n\left(\epsilon - \left(2\left(\frac{\epsilon^2}{1-\epsilon}\right) - 1\left(\frac{\epsilon^3}{(1-\epsilon)^2}\right)\right)\right)$$

errors. In the next iteration we have at most

$$2n\left(\epsilon - 3\left(\frac{\epsilon^2}{(1-\epsilon)}\right) + 3\left(\frac{\epsilon^3}{(1-\epsilon)^2}\right) - \left(\frac{\epsilon^4}{(1-\epsilon)^3}\right)\right)$$

errors. Of course, one can observe the binomial coefficients here.

Back-Track makes no nut-and-bolt comparisons, but it does depend on the particular comparisons that were made before by Find_Misplaced_Elements and Get-c-Approximate-Median.

The algorithm Back-Track only returns a "few" additional elements to the active list X_i (a few relative to the size of X_i). The algorithm Find_Misplaced_Elements is applied to geometrically smaller fringes each time and for each application to one of these fringes with n elements it does less than cn additional nut-and-bolt comparisons, for some constant c. Hence, it does not change the asymptotic number of nut-and-bolt comparisons of the algorithm Get-c-Approximate-Median.

In particular, suppose that we ϵ-halve on a fringe until we have the list $X_k^F[n+1,(K\epsilon+1)n]$, where $k = \lceil -\lg(K\epsilon)\rceil$. (Note that $\epsilon < \frac{1}{K}$.) Suppose we ϵ-halve the side containing the potential extraneous elements.

```
Find_Misplaced_Elements($X, i, m$)
    $r \leftarrow |X|$;    $\ell \leftarrow 1$
    if $i$ is odd, then $Z_1 \leftarrow X[(\ell+r)/2,\ r]$
        else $Z_1 \leftarrow X[\ell,\ (\ell+r)/2]$
    $j \leftarrow 1$
    While $|Z_j| \geq K\epsilon m$ do
        $Z_j \leftarrow$ nut-and-bolt-$\epsilon$-halve($Z_j[\ell, r]$)
        if $i$ is odd, then        (* $i$ determines the side we come from *)
            $r \leftarrow r - (\ell+r)/2$      (* Shrink the right boundary *)
        else
            $\ell \leftarrow \ell + (\ell+r)/2$      (* Shrink the left boundary *)
        $Z_{j+1} \leftarrow Z_j[\ell, r]$       (* Sliding over *)
        $j \leftarrow j+1$
    od
Return $Z_j$
```

Figure 3: Finding Misplaced Nuts and Bolts

In the most general terms the basic idea behind the algorithm Find_Misplaced_Elements is that ϵ-halving the high elements of X_i in the right fringe towards its boundary with X_{i+1} will bring most of the misplaced, if any, smaller elements toward this fringe's boundary with the active list X_{i+1}. This is because the extraneous elements from $\mathcal{S}_{n/2}$ are smaller than all other elements in $X[n+1, 2n]$. The symmetric case occurs for the left fringe.

Furthermore, we keep ϵ-halving in order to allow the (too high) extraneous elements from subsequent iterations of Find_Misplaced_Elements to be pushed up exponentially fast and left behind as we continue to ϵ-halve towards the left boundary of the right fringe. We only ϵ-halve the right fringe until we have an array $X_k^F[n+1,\ (K\epsilon+1)n]$.

If in the first run of Find_Misplaced_Elements from $X^F[n+1, 2n]$ down to $X_k^F[n+1,\ (K\epsilon+1)n]$, where, for example $K \leq 32$ so X_k is of size at least $32\epsilon n - 1$, so we must have at least $24\epsilon n - 1$

```
Back-Track($Y, i, Z$)
    if $i \leq 2$ then Return $\emptyset$
    if $i$ is even, then
        for any members of $Z$ that are in the right half of $Y$
        find all members of all of the left fringes that are supported exclusively
        by these members of $Z$ or other active elements. Put these candidate illicitly
        supported elements in $B$.
    else
        for any members of $Z$ that are in the left half of $Y$
        find all members of all of the right fringes that are supported exclusively
        by these members of $Z$ or other active elements. Put these candidate illicitly
        supported elements in $B$.
Return $B$
```

Figure 4: Back-Tracking

comparable elements since we can loose at most

$$2\epsilon n \sum_{i=0} \frac{1}{2^i} \quad \leq \quad 4\epsilon n \tag{6}$$

elements total from all of the ϵ-halving in Find_Misplaced_Elements. But, we know that $4\epsilon n$ is $\frac{1}{8}$ of the total $32\epsilon n$ elements that remain, hence by Lemma 3 we can continue ϵ-halving at least up to this point.

Lemma 5 Provided that we can ϵ-halve in the i-th iteration, then the list $X[\ell(i),\ r(i)]$ has fewer than

$$\varphi_i \quad = \quad 2.5n \left[\frac{\epsilon}{2^i} + \frac{\epsilon^2}{2^{i-1}} + \frac{\epsilon^3}{2^{i-2}} + \cdots + \frac{\epsilon^i}{1} \right] \tag{7}$$

elements without matches after the i-th iteration of Get-c-Approximate-Median.

Proof: This follows directly from Lemma 3, while here we over-estimate the number of extraneous elements in the list $X[\ell(i),\ r(i)]$ from the previous iterations. We add the 0.5 to cover the extra overhead due to the $K\epsilon n/2^i$ nuts and bolts added back in the list in the i-th iteration and it also bounds the variable number of elements added back by Back-Track in this iteration. ∎

As the leading coefficient for φ_i we just write 2 instead of 2.5 from here on for ease of exposition.

This means, $\varphi_0 = 2\epsilon n$, $\varphi_1 = \epsilon n + 2\epsilon^2 n$, $\varphi_2 = \epsilon n/2 + \epsilon^2 n + 2\epsilon^3 n$, etc. This represents an over-estimate of the diminishing number of errors that can come about from the ϵ-halving, see Lemma 4.

Notice that

$$\left[\frac{\epsilon}{2^i} + \frac{\epsilon^2}{2^{i-1}} + \frac{\epsilon^3}{2^{i-2}} + \cdots + \frac{\epsilon^i}{1} \right] \quad \leq \quad \left(\frac{\epsilon}{2^{i-1}} \right) \tag{8}$$

so that φ_i is a small fraction of n depending on i.

We now go on to show that we can continue ϵ-halving as our algorithm iterates. To do this, we carefully track the effect of Back-Track and Find_Misplaced_Elements. To this end, take the next definition of ϕ_i.

$$\phi_i \;=\; \varphi_i \left(\sum_{j=0}^{\lceil -\lg K\epsilon \rceil} \frac{\epsilon^j}{(1-\epsilon)^{\max\{j-1,0\}}} \binom{\lceil -\lg K\epsilon \rceil}{j} (-1)^j \right).$$

We keep in mind that Find_Misplaced_Elements does not retrieve any elements until the start of the 2-nd iteration of Get-c-Approximate-Median.

Assuming there are as many errors as possible by the ϵ-halving, then ϕ_i is an upper bound on the number of active elements that are lost to a fringe and that Find_Misplaced_Elements gets back.

The term

$$\sum_{j=0}^{\lceil -\lg(K\epsilon) \rceil} \frac{\epsilon^j}{(1-\epsilon)^{\max\{j-1,0\}}} \binom{\lceil -\lg K\epsilon \rceil}{j} (-1)^j$$

follows directly by induction as a consequence of Find_Misplaced_Elements. This comes from the fact that we must subtract off the errors that accumulate in the successive iterations of Find_Misplaced_Elements on a particular fringe. Further, the φ_i comes directly from Lemma 5. But intuitively the value of φ_i is due to the fact that the "extreme errors," which are from older time steps, are pushed away faster than the less extreme errors, which are from more recent time steps. See also Lemma 4.

Lemma 6 Let φ_i be as defined in Equation 7 and suppose i is odd. Given the list $X[\ell(i),\ r(i)]$ immediately after the i-th iteration of Get-c-Approximate-Median and suppose that we just lost φ_i elements to the right fringe and all of these φ_i elements are active in $X[\ell(i+2),\ r(i+2)]$; then in the one run of Find_Misplaced_Elements on the right fringe the list $X[\ell(i+2),\ r(i+2)]$ gets back more than $3\varphi_i/4$ of the elements lost to the right fringe in the i-th iteration.

Proof: Using the algorithm Find_Misplaced_Elements we will gain back at least ϕ_i of these original φ_i lost elements. For very small, but constant, ϵ we can show $\phi_i \geq 3\varphi_i/4$ completing the proof. ▮

In the $(i+2)$-nd iteration, at least $3\varphi_i/4 \geq \varphi_{i+2}$ elements will be returned to the list $X[\ell(i+2),\ r(i+2)]$ for subsequent ϵ-halving. Since $3\varphi_i/4 \geq \varphi_{i+2}$ more active elements can be returned than can be lost in iteration $i+2$ in Get-c-Approximate-Median.

We may loose many elements once they become in-active, but this does not effect our algorithm in an adverse way. That is, once some elements become in-active, then they cannot be illicitly supported any more, hence back-tracking will not find them.

Lemma 7 If a set U of elements in the first fringe are all active for i iterations, then at each iteration each element of this set must be illicitly supported (directly in iteration 1, and indirectly there after) only by active elements.

Proof: The proof follows directly by induction. ▮

Furthermore, we also back-track and find all elements we can, which have illicit supports in the present list under consideration. By Lemma 7, we know that as we decrease the size of the list X_i under consideration, we also decrease the size of the errors.

Lemma 8 Suppose that we loose at most

$$\psi_i \;=\; 2\epsilon n/2^i - \phi_i$$

elements to the outer left fringe in step i of Get-c-Approximate-Median, then in $j + \lceil -\lg \epsilon \rceil$ more iterations we will gain back via Back-Track at least

$$\psi_i \sum_{k=1}^{j} \frac{1}{2^k}$$

still active elements from this outer fringe.

Proof: By Lemma 7, we consider only active elements which are directly or indirectly supported. From here, we discard the binomial coefficients (of $\lceil -\lg K\epsilon \rceil$) since they are due to the $\lceil -\lg K\epsilon \rceil$ levels in the outer fringe. Any illicitly supported elements in each level have the same amount of bounded support.

Now it is important to note that each illicitly supported element in the outer fringes is illicitly supported by only a constant number of comparisons that were defined by a nut and bolt ϵ-halving network.

Note that the leading coefficient of the term φ_i is not greater than $2\epsilon n/2^i$. Now, consider ψ_i-s higher order term $\epsilon^2 n/2^{i-1}$ (which is divided by the leading binomial coefficient of $\lceil -\lg K\epsilon \rceil$) and we know that in $i + \lceil -\lg \epsilon \rceil$ iterations the leading error term of $X_{i+\lceil -\lg \epsilon \rceil}$ is at most

$$\begin{aligned} \varphi_{i+\lceil -\lg \epsilon \rceil} &\leq \epsilon n/2^{i+\lceil -\lg \epsilon \rceil} \\ &\leq \epsilon^2 n/2^i \end{aligned}$$

hence since each illicitly supported active element q, if q is not returned by Back-Track, then q must have at least one illicit support that is in error (that is, an element not in the active list $X_{i+\lceil -\lg \epsilon \rceil}$, but this element "belongs" in $X_{i+\lceil -\lg \epsilon \rceil}$). Recall that each element has only a constant number of supports in any ϵ-halver. However, by the $(i + \lceil -\lg \epsilon \rceil + j)$-th iteration we know that

$$\begin{aligned} \varphi_{i+\lceil -\lg \epsilon \rceil + j} &\leq \epsilon^2 n/2^{i+j} \\ &\leq \epsilon^2 n/2^{i-1}. \end{aligned}$$

Now, since the leading term of ψ_i (that is, the number of elements that are potentially illicitly supported in the i-th left outer fringe) divided by $\lceil -\lg K\epsilon \rceil$ is $\epsilon^2 n/2^{i-1}$ we know that at most $\epsilon^2 n/2^{i-1}$ elements can still be illicitly supported in the i-th left outer fringe. This is because, for each illicitly supported element q to remain illicitly supported it must have at least one active element q' that (incorrectly) remains in some fringe. Otherwise, it will be found by Back-Track. Furthermore, each subsequent iteration of Get-c-Approximate-Median reduces the error term geometrically giving the statement of the lemma, see also Equation 8. ∎

The proof of Lemma 8 indicates that Back-Track may return variable sized sets. However, over many iterations of Get-c-Approximate-Median (subsequences of $\lceil -\lg \epsilon \rceil$ iterations) the cardinality of these sets diminishes geometrically.

The following theorem shows that as our algorithm iterates it maintains a steady-state in the proportion of matching nuts and bolts in the active list.

Theorem 5 Let $i : \lceil \lg n \rceil \geq i \geq 2$ and let c' be a constant. After each iteration i Get-c-Approximate-Median maintains at least

$$n/2^i - c'\varphi_{i-2} \quad > \quad \left(1 - \frac{1}{8}\right) n/2^i$$

nuts with matching bolts in X_i.

Proof: This is by the number of active elements that remain misplaced from ϵ-halving and the number of elements gained back by running Find_Misplaced_Elements and Back-Track. Also, note that $\varphi_i = \phi_i + \psi_i$ and $2\phi_i + 2\psi_i = 2\varphi_i \leq 4\epsilon n/2^i$ holds by definition.

Now we must also consider the number of elements that already have been lost by the time we get to iteration i of Get-c-Approximate-Median. By Lemma 8, within a constant number ($\lceil -\lg \epsilon \rceil + 1$) of iterations we will have gained back at least $(\psi_{i+\lceil -\lg \epsilon \rceil})/2$ of potential active elements lost to the outer left fringe. Furthermore, we will gain back a geometrically larger proportion of elements lost to older outer fringes. Also, by Lemma 6, we gain ϕ_i elements back from the right (left) fringes by Find_Misplaced_Elements in the i-th iteration. Finally, for some constant c', the next inequality holds since $\lceil -\lg \epsilon \rceil$ is a constant,

$$\sum_{k=i-3}^{i-\lceil -\lg \epsilon \rceil} \varphi_k \quad \leq \quad c'\varphi_{i-2}.$$

For sufficiently small ϵ we have $c'\epsilon < \frac{1}{8}$.

By Lemma 8, at least $(\psi_{i-\lceil -\lg \epsilon \rceil -1})/2$ active elements lost to the $(i - \lceil -\lg \epsilon \rceil - 1)$-th outer fringe are returned by Back-Track by iteration i. Also, we know that,

$$(\psi_{i-\lceil -\lg \epsilon \rceil -1})/2 \quad \geq \quad \frac{\left(\sum_{k=i-\lceil -\lg \epsilon \rceil}^{i-3} \psi_k \right)}{4}.$$

In the next iteration, at least $(\psi_{i-\lceil -\lg \epsilon \rceil -1})/4$ more active elements are returned from the $(i - \lceil -\lg \epsilon \rceil - 1)$-th outer fringe by Back-Track. Considering the values ϕ_i returned by Find_Misplaced_Elements, we see that we will always have less than $c'\varphi_i$ active elements missing from the i-th iteration.

In other words, as Get-c-Approximate-Median runs up through iteration i it has lost at most a bounded number of active elements while within a constant number of iterations the number of active elements recovered by Find_Misplaced_Elements from the most recent fringes and recovered by Back-Track from all of the outer fringes is an appreciable fraction of the number of elements lost to the outer fringe.

Furthermore, for an appropriate choice of ϵ, we know that $c'\epsilon \leq \frac{1}{8}$, hence we can always continue ϵ-halving by Lemma 3. Also, for suitable choices for our constants if we can ϵ-halve X_i, then we can always ϵ-halve X_i's fringes. ▮

Theorem 5 shows that Lemma 3 can be applied. This allows us to continually ϵ-halve the present lists under consideration.

The next theorem follows from the results of this section. In particular, it follows from Lemma 4 and Theorem 5.

Theorem 6 The algorithm Get-c-Approximate-Median maintains Invariant 1 through iteration $t \leq \lceil \lg n \rceil$ where $|X_t| \geq C$, for C a suitably large constant, and further this algorithm takes $O(n)$ comparisons of nuts and bolts.

The constant C depends on the size of ϵ and the constraints given in the results in this section. Just the same, the constant K depends on c' and ϵ. We can choose K large enough so that the term $c'\varphi_i$ will allow $\lceil -\lg K\epsilon \rceil$ ϵ-halvings of the fringes.

Suppose $|X_i|$ is of constant cardinality, for some i; then we know that Invariant 1 holds, hence we can check all of the elements of X_i to find which are a c-approximate medians.

Theorem 6 shows that Get-c-Approximate-Median produces a c-approximate median. Using this c-approximate median we can split the list into two halves with all matching elements. Hence, we can continue the same procedure. This leads directly to the existence of the $O(n \lg n)$ nut-and-bolt matching algorithm as discussed in the beginning of this paper.

4 Conclusions

This paper shows the existence of an algorithm for solving the nuts and bolts problem in $O(n \lg n)$ nut-and-bolt matching operations. There are huge constants hidden in the asymptotic notation here, though we don't give them explicitly. Reducing these constants (perhaps by removing the expanders) would be interesting.

Also, it would be interesting to try out the nuts-bolts and washers matching problem and other obvious generalizations of the nuts and bolts problem.

5 Acknowledgments

A special thanks to Rudolf Fleischer for many helpful discussions and comments. Thanks to Andrea Rafael, Jesper Träff, Vasilis Capoyleas, and Shiva Chaudhuri for reading and commenting on versions of this paper. Thanks to Gregory Rawlins for suggesting the nuts-and-bolts problem as an area of research. And thanks to Michiel Smid and Torben Hagerup for several discussions.

References

[1] M. Ajtai, J. Komlós, W. L. Steiger, and E. Szemerédi: "Deterministic Selection in $O(\log \log n)$ Parallel Time," In *Proceedings of the Symposium on the Theory of Computing*, (STOC), ACM-Press, 188-195, 1986.

[2] M. Ajtai, J. Komlós, and E. Szemerédi: "An $O(n \log n)$ Sorting Network," In *Proceedings of the Symposium on the Theory of Computing*, (STOC), ACM-Press, 1-9, 1983.

[3] M. Ajtai, J. Komlós, and E. Szemerédi: "Sorting in $c \log n$ Steps," *Combinatorica*, **3(1)**, 1-19, 1983.

[4] N. Alon: "Eigenvalues and Expanders," *Combinatorica*, **6(2)**, 83-96, 1986.

[5] N. Alon and Y. Azar: "Finding an Approximate Maximum," *SIAM J. on Computing*, **18(2)**, 258-267, 1989.

[6] N. Alon, M. Blum, A. Fiat, S. Kannan, M. Naor, and R. Ostrovsky: "Matching Nuts and Bolts," *Proceedings of the 5-th Annual Symposium on Discrete Algorithms* (SODA '94), ACM-SIAM Press, 690-696, 1994.

[7] P. G. Bradford and R. Fleischer: "Matching Nuts and Bolts Faster," *Max-Planck-Institut für Informatik, Technical Report* MPI-I-95-1-003, Saarbrücken, Germany, May 1995. Also, an updated version to appear in the proceedings of the *Sixth International Symposium on Algorithms and Computation (ISAAC '95).*

[8] T. H. Cormen, C. E. Leiserson, and R. L. Rivest: *Introduction to Algorithms*, McGraw Hill/MIT Press, 1990.

[9] F. R. K. Chung: "On Concentrators, Superconcentrators, generalizers, and nonblocking networks," *Bell System Tech. J.*, **58**, 1765-1777, 1978.

[10] W. Goddard, C. Kenyon, V. King, and L. Schulman: "Optimal randomized algorithms for local sorting and set-maxima," *SIAM J. on Computing*, **22**, 272-283, 1993.

[11] D. E. Knuth: *The Art of Computer Programming*, Vol. 3: *Searching and Sorting*, Addision-Wesley, 1973.

[12] J. Komlós, Y. Ma, and E. Szemerédi: "Matching Nuts and Bolts in $O(n \log n)$ Time," to appear in *SODA '96.*

[13] A. Lubotzky: *Discrete Groups, Expanding Graphs and Invariant Measures*, Birkhäuser, Progress in Mathematics Series # 125, 1994.

[14] J. I. Munro and M. Paterson: "Selection and Sorting with Limited Storage," *Theoretical Computer Science*, **12**, 315-323, 1980.

[15] I. Parberry: *Parallel Complexity Theory,* Research Notes in Theoretical Computer Science, Pitman Publishing Co., 1987.

[16] M. S. Paterson: "Improved Sorting Networks with $O(\log N)$ Depth," *Algorithmica*, **5**, 75-92, 1990.

[17] M. Pinsker: "On the Complexity of a Concentrator," In *Proceedings of the 7-th International Teletrafic Conference*, Stockholm, June 1973.

[18] N. Pippenger: "Superconcentrators," *SIAM J. on Comp.*, **6(2)**, 298-304, June 1977.

[19] N. Pippenger: "Sorting and Selecting in Rounds," *SIAM J. on Comp.*, **16(6)**, 1032-1038, 1987.

[20] G. J. E. Rawlins: *Compared To What? An Introduction to the Analysis of Algorithms*, W. H. Freeman/Computer Science Press, 1992.

[21] P. Sarnak: *Some Applications of Modular Forms*, Cambridge tracts in mathematics # 99, Cambridge University Press, 1990.