Matching nuts and bolts

(Extended Abstract)*

Noga Alon † Manuel Blum ‡ Amos Fiat § Sampath Kannan ¶
Moni Naor‖ Rafail Ostrovsky **

Abstract

We describe a procedure which may be helpful to any disorganized carpenter who has a mixed pile of bolts and nuts and wants to find the corresponding pairs of bolts and nuts. The procedure uses our (and the carpenter's) ability to construct efficiently highly expanding graphs. The problem considered is given a collection of n bolts of distinct widths and n nuts such that there is a 1-1 correspondence between the nuts and bolts. The goal is to find for each bolt its corresponding nut by comparing nuts to bolts but not nuts to nuts or bolts to bolts. Our objective is to minimize the number of operations of this kind (as well as the total running time). The problem has a randomized algorithm similar to Quicksort. Our main result is an $n(\log n)^{O(1)}$-time *deterministic* algorithm, based on expander graphs, for matching the bolts and the nuts.

1 Introduction

Given a collection of n bolts of pairwise distinct widths and n corresponding nuts, our objective is to find for each bolt its corresponding nut. By trying to match a bolt and a nut we can see which one is bigger, and our aim is to minimize the number of operations of this kind (as well as the total running time of the rest of the algorithm). Note that we are not allowed to compare two bolts or two nuts directly. The mathematical description of the problem is thus the following; 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 widths 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 nuts and bolts matching problem is first mentioned as an exercise in [14], page 293. There is a simple randomized algorithm along the lines of Quicksort for this problem and this solution is described later in this section.

Since there are $n!$ possibilities for σ, the obvious information theoretic lower bound shows

*To appear in Fifth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 94) January 23-25, 1994, Arlington, Virginia.

†Department of Mathematics, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv, Israel and AT & T Bell Labs, Murray Hill, NJ 07974, USA. e-mail: noga@math.tau.ac.il. Research supported in part by a United States Israel BSF Grant

‡Computer Science Division, University of California at Berkeley, Berkeley, CA 94720, USA. e-mail: blum@cs.berkeley.edu. Supported by NSF grant CCR92-01092.

§Department of Computer Science, Raymond and Beverly Sackler Faculty of Exact Sciences, Tel Aviv University, Tel Aviv, Israel. e-mail: fiat@math.tau.ac.il. Research supported by a grant from the Israeli Academy of Sciences

¶Department of Computer Science, University of Arizona, USA. e-mail: kannan@cs.arizona.edu.

‖Department of Applied Mathematics and Computer Science, The Weizmann Institute of Science, Rehovot 76100, Israel. e-mail: naor@wisdom.weizmann.ac.il. Supported by an Alon Fellowship.

**University of California at Berkeley Computer Science Division, and International Computer Science Institute at Berkeley. e-mail: rafail@melody.berkeley.edu. Supported by NSF postdoctoral fellowship and ICSI. Part of this work was done while visiting Tel Aviv University and Hebrew University of Jerusalem.

that any bounded degree decision tree that solves the problem has depth at least $\log(n!) = \Theta(n \log n)$. In particular, at least $\Omega(n \log n)$ comparisons are needed. This is a lower bound for the expected number of comparisons in any randomized algorithm for the problem as well.

A simple modification of Quicksort shows that there are randomized algorithms whose expected number of comparisons (and running time) are $O(n \log n)$: pick a random bolt, compare it to all the nuts, find its matching nut and compare it to all the bolts, thus splitting the problem into two problems, one consisting of the nuts and bolts smaller than the matched pair and one consisting of the larger ones. Repeating in this manner yields an algorithm whose expected running time can be analyzed by imitating the known analysis for Quicksort (see, e.g., [8]) showing that it is $\Theta(n \log n)$. Moreover, it is easy to modify the above algorithm and make sure that the probability its running time will considerably exceed its expectation will be exponentially small.

Deterministic algorithms seem more difficult to find. In fact, even obtaining an $o(n^2)$ algorithm appears to be a non-trivial task. We have two different approaches for the problem. If we count only comparisons, and allow ourselves to deduce implications by transitivity for free, then we can apply some of the techniques used in the study of parallel approximate sorting and selecting comparison algorithms (see [1], [2], [3] [7] [13]) and obtain deterministic, explicit algorithms with $O(n(\log n)^{3+\epsilon})$ comparisons. Moreover, these algorithms can be parallelized and we can also show that the minimum number of comparisons needed in a k-round algorithm is $\tilde{\Theta}(n^{1+1/k})$, where here we use the common $f = \tilde{\Theta}(g)$ notation to indicate that f and g are equal up to polylogarithmic factors. More details will be given in the final version of this paper. Let us just remark that the $\Omega(n^{1+1/k})$ lower bound for the number of comparisons in any (deterministic or randomized) k-round algorithm follows by counting from the well known fact that the number of acyclic orientations of any graph with m edges and n vertices is at most $((2m+n)/n)^n$.

The problem becomes harder if one is interested in a "real" algorithm, i.e. we count both the number of comparisons and the time to decide which comparisons to perform. Our main result is an $n(\log n)^{O(1)}$-time algorithm for matching nuts and bolts. The algorithm constructs "deterministic samples" by applying expander graphs in an interesting way.

An outline of the algorithm is given in the next section and a detailed description together with a proof of correctness are given in Section 3.

2 Outline of the algorithm

Our idea is to try to find a good pivot for the partitioning in the Quicksort like algorithm described above. For a nut $s \in S$ define $rank(s)$ as $|\{s' \in S : s \geq s'\}|$. The rank of a bolt $b \in B$ is defined similarly. A good pivot is a nut whose rank is roughly $n/2$. To decide whether a nut is a good pivot we can take a sample of the bolts and compare the nut to all the bolts in the sample. As a first try we associate with every nut s a sample T_s of bolts. The algorithm is as follows:

1. For every nut s compare s to all the bolts in T_s.

2. Delete from further consideration all the nuts s where s is not larger than roughly half of T_s and smaller than roughly half of T_s.

3. Start exhaustively testing the remaining nuts until you find one that is a good pivot.

The number of comparisons such an algorithm requires is the sum of the sizes of the samples, plus n times the number of bad pivots remaining at Step 3 of the algorithm. One can show that constructions for the samples T_s exist where the size of each T_s is $\sqrt{n \log n}$, and

at Step 3 of the algorithm at most $\sqrt{n \log n}$ bad pivots remain. Thus the total complexity is $O(n^{1.5}\sqrt{\log n})$. As for explicit constructions, one can choose the samples T_s according to a projective plane and get an explicit $O(n^c)$-time algorithm for some $c < 2$. We omit the details.

However, $O(n^{1.5})$ seems to be the limit of any such scheme, regardless of the construction of the samples. To get down to $O(n \log n^{O(1)})$ we need a more sophisticated algorithm.

Our scheme consists of $\ell = \log_2 n$ iterations where in each iteration half the nuts survive for the next one. We associate with every nut $s \in S$ a sequence of ℓ samples $T_s^0, T_s^1, \ldots T_s^\ell$ where the size of T_s^i is t_i for $0 \leq i \leq \ell$. The t_i's increase with i. Our algorithm starts with a pile of nuts and a pile of bolts. Over time it discards nuts until it has a single nut remaining. The set of nuts that survived till the ith iteration is denoted by S_i.

For $i = 0$ to ℓ

1. For every remaining nut $s \in S_i$ compare s to all the bolts in T_s^i.

2. For every remaining nut $s \in S_i$ see how close to $t_i/2$ is the rank of s in T_s^i.

3. Delete from further consideration half the nuts, those whose ranks are the furthest from $t_i/2$.

The remaining nut is our candidate for a good pivot.

The complexity of the algorithm is now

$$\sum_{i=0}^{\ell} n/2^i \cdot t_i.$$

Therefore t_i can double in each iteration without effecting the total complexity by much. What we still must specify is:

1. An efficient construction for the samples T_s^i.

2. The relationship between the nuts in S_i and in S_{i+1}. This relationship should somehow allow us to deduce that the surviving element after ℓ iterations is a good pivot.

The samples T_s^i's are constructed via expander graphs. We identify the nuts with the nodes of such a graph, associate every surviving nut s with a disjoint set of 2^i nodes and take T_s to be the set of all neighbors of the set associated with s.

The property that the surviving nuts S_i should maintain is that most of them are good pivots, however the definition of a good pivot should relax somewhat in time,since we should leave room for errors from the sampling. The exact property we will maintain is that for half the elements of S_i their rank among all the bolts is between $\frac{n}{4} - i \cdot c$ and $\frac{3n}{4} + i \cdot c$, where c is some number smaller than, say, $n/8 \log n$. It is easy to verify that if S_ℓ maintains this property, then its only surviving member is a good pivot.

3 An $O(n(\log n)^4)$ algorithm

We now provide a detailed description of the algorithm and the proof of correctness. Subsection 3.1 introduces the main tool our carpenter uses, the expander graph, and shows how to construct the required samples from it. Subsection 3.2 gives the main Lemma on expanders which is used in the proof of correctness in Subsection 3.3.

3.1 The algorithm

For a nut $s \in S$ define $rank(s)$ by $rank(s) = |\{s' \in S : s \geq s'\}|$. The rank of a bolt $b \in B$ is defined similarly. A nut s is called *an approximate median* if $n/10 \leq rank(s) \leq 9n/10$. As described above, an approximate median is a good pivot: given an approximate median s we can compare it to all bolts, find its matching bolt b, compare b to all nuts and split the problem into two subproblems of the same type, each of size between $n/10$ and $9n/10$. Therefore, in order to get the required $O(n(\log n)^4)$ algorithm it suffices to design an $O(n(\log n)^3)$ algorithm for finding an

approximate median.

Our algorithm for finding an approximate median uses a sequence of bipartite (multi-) graphs H_i. All these graphs will be constructed from a single expander. Let $G = (V, E)$ be a d-regular graph on a set $V = \{v_1, \ldots, v_n\}$ of n vertices in which the absolute value of all nontrivial eigenvalues is at most $2\sqrt{d-1}$. There are known explicit constructions of such graphs (see [10], [11]) for any d for which $d-1$ is a prime power congruent to 1 modulo 4, where for each such d there are constructions for an infinite set of values of n containing a number between x and cx for all large x, where c is some absolute constant. By adding dummy bolts and nuts, if necessary, we may assume, thus, that a graph G as above exists, where n is the number of bolts (and nuts). Moreover, the known constructions of these graphs enable one to construct them in time proportional to their number of edges. For our algorithm we take $d = (1+o(1))10^8 \log_2^2 n$, where here, and in what follows, we make no attempt to optimize the multiplicative constants.

For each i, $0 \leq i \leq \log_2 n$, define a bipartite graph H_i with classes of vertices U and W, where $|U| = \lfloor n/2^i \rfloor$ and $|W| = n$ as follows. Put $I = \lfloor n/2^i \rfloor$ and let $V = V_1 \cup V_2 \cup \ldots \cup V_I$ be an arbitrary partition of V into I almost equal pairwise disjoint parts. Thus $n/2I \leq \lfloor n/I \rfloor \leq |V_j| \leq \lceil n/I \rceil \leq 2n/I$ for all j. Denote $U = \{u_1, \ldots, u_I\}$ and $W = \{w_1, \ldots, w_n\}$. The number of parallel edges between u_j and w_k is simply the number of neighbors of v_k in V_j in the original graph G.

The algorithm consists of $\ell = \lfloor \log_2 n \rfloor$ iterations. In the beginning of iteration number i, $(0 \leq i < \lfloor \log_2 n \rfloor)$ we have a subset S_i of cardinality $|S_i| = I = \lfloor n/2^i \rfloor$ of the set of nuts. (For $i = 0$, $S_0 = S$ is the set of all nuts, and each S_i will be a subset of S_{i-1} defined as described below). Suppose $S_i = \{s_1, \ldots, s_I\}$ and let $B = \{b_1, \ldots, b_n\}$ be the set of all bolts. In the ith iteration, we compare s_j to b_k for every edge $u_j w_k$ of H_i. For each nut $s_j \in S_i$, define the outdegree $outdeg(s_j)$ to be the number of comparisons as above for which s_j turned out to be at least as big as b_k. Note that since the graphs H_i may have parallel edges, a comparison may contribute more than 1 to such an outdegree. The normalized outdegree $ndeg(s_j)$ is now defined as $outdeg(s_j)/deg(u_j)$, where $deg(u_j)$ $(= d|V_j|)$ is the total degree of u_j in the graph H_i. Intuitively, $ndeg(s_j)n$ gives an approximation to the actual rank of s_j. The set S_{i+1} is now chosen as the subset of those $\lfloor n/2^{i+1} \rfloor$ elements s_j of S_i whose normalized outdegrees are closest to $1/2$ (equalities are broken arbitrarily). The algorithm ends after $i = \lfloor \log_2 n \rfloor$ iterations with a set S_i of one element. As we show in the next subsection this element is an approximate median, i.e. a nut whose rank is between $n/10$ and $9n/10$.

3.2 A lemma on expanders

We now state and prove the technical lemma required to show that the algorithm described above works. It essentially says that for any expander and any partition of its vertices, every subset of the vertices A has the property that the fraction of parts for which the number of edges between A and the part deviates significantly from the "expectation" is small.

LEMMA 3.1. *Let $G = (V, E)$ be a d-regular graph on n vertices and suppose that the absolute value of each nontrivial eigenvalue of G is at most λ. Suppose $I \leq n$, and let $V = V_1 \cup V_2 \cup \ldots \cup V_I$ be a partition of V into I pairwise disjoint sets, so that $n/2I \leq |V_j| \leq 2n/I$ for all j. Let A be a subset of V, and let $e(V_j, A)$ denote the total number of ordered pairs (v_j, a), with $v \in V_j$ and $a \in A$ such that $v_j a$ is an edge of G. Define*

$$l = |\{j : \ |e(V_j, A) - |A||V_j|d/n| \geq \epsilon |V_j| d\}|.$$

Then

$$\frac{l}{I} \leq \frac{8\lambda^2 |A|}{d^2 \epsilon^2 n}.$$

In particular, if $\lambda \leq 2\sqrt{d-1}$ *and* $\epsilon = \frac{100}{\sqrt{d}}$, *then* $l/I < 1/200$.

Proof. Let $n(v) = n_A(v)$ denote the number of neighbors of v in A. By [4] (see also [5], page 122),

$$\sum_{v \in V} (n(v) - |A|d/n)^2 \leq \lambda^2 |A|(1 - |A|/n) \leq \lambda^2 |A|.$$

By the Cauchy-Schwartz Inequality, for each fixed j,

$$\begin{aligned} &\frac{1}{|V_j|}(e(V_j, A) - |A|d|V_j|/n)^2 \\ &= \frac{1}{|V_j|}(\sum_{v \in V_j} (n(v) - |A|d/n))^2 \\ &\leq \sum_{v \in V_j} (n(v) - |A|d/n)^2. \end{aligned}$$

Since $|V_j| \leq 2n/I$ for all j this implies that

$$\begin{aligned} &\frac{I}{2n} \sum_{j=1}^{I} (e(V_j, A) - |A|d|V_j|/n)^2 \\ &\leq \sum_{j=1}^{I} \sum_{v \in V_j} (n(v) - |A|d/n)^2 \\ &= \sum_{v \in V} (n(v) - |A|d/n)^2 \leq \lambda^2 |A|. \end{aligned}$$

By the definition of l, and since $|V_j| \geq n/2I$ for all j, we conclude that

$$\frac{I}{2n} l \epsilon^2 \frac{n^2}{4I^2} d^2 \leq \lambda^2 |A|,$$

implying the desired upper bound for l. □

3.3 The proof of correctness

The discussion at the end of Section 2 implies that in order to prove that the algorithm in Subsection 3.1 indeed finds an approximate median it is sufficient to show the following.

CLAIM 3.1. *For each* i, *the ranks of at least half of the elements in* S_i *are between* $\frac{n}{4} - 200\frac{in}{\sqrt{d}}$ *and* $\frac{3n}{4} + 200\frac{in}{\sqrt{d}}$.

Proof: We apply induction on i. The result clearly holds for $i = 0$. Assuming it holds for i we prove it for $i+1$. Let S_i^{good} be the set of all members of S_i whose ranks r satisfy

$$\frac{n}{4} - 200\frac{in}{\sqrt{d}} \leq r \leq \frac{3n}{4} + 200\frac{in}{\sqrt{d}}.$$

Let S_i^{medium} be the set of all members of S_i whose ranks r satisfy

$$\frac{n}{4} - 200\frac{(i+1)n}{\sqrt{d}} \leq r < \frac{n}{4} - 200\frac{in}{\sqrt{d}}$$

or

$$\frac{3n}{4} + 200\frac{in}{\sqrt{d}} < r \leq \frac{3n}{4} + 200\frac{(i+1)n}{\sqrt{d}},$$

and let S_i^{bad} be all the other members of S_i. We must show that it is impossible that more than half of the members of S_{i+1} will come from S_i^{bad}. Assume this is the case. Then there is a subset $T \subset S_i^{good}$ and a subset $Z \subset S_i^{bad}$, so that $|T| = |Z| \geq |S_i|/4$ and in the ith iteration, the normalized outdegree of every $z \in Z$ was closer to $1/2$ than the normalized outdegree of every $t \in T$. We show that this is impossible by applying Lemma 3.1.

Indeed, by this lemma, with A being the set of all vertices in W corresponding in the graph H_i to bolts whose ranks are below $n/4 - 200ni/\sqrt{d}$ and with $\epsilon = 100/\sqrt{d}$ we conclude that all but at most $|S_i|/200$ members of S_i^{good} have normalized outdegree strictly bigger than $1/4 - 200i/\sqrt{d} - 100/\sqrt{d}$. Similarly, by applying this lemma with A being the set of all vertices corresponding to bolts of ranks above $3n/4 + 200ni/\sqrt{d}$ we conclude that all but at most $|S_i|/200$ members of S_i^{good} have normalized outdegree strictly less than $3/4 + 200i/\sqrt{d} + 100/\sqrt{d}$. Thus, the normalized outdegrees of all but at most $0.01|S_i|$ members of S_i^{good} are between these two bounds.

A similar application of the lemma with A being the set of all vertices corresponding to bolts of ranks less than $n/4 - 200n(i+1)/\sqrt{d}$

(as well as to the symmetric set corresponding to bolts of ranks greater than $3n/4 + 200n(i+1)/\sqrt{d}$) shows that all but at most $0.01|S_i|$ members of S_i^{bad} have normalized outdegrees which are either smaller than $1/4 - 200(i+1)/\sqrt{d} + 100/\sqrt{d}$ or bigger than $3/4 + 200(i+1)/\sqrt{d} - 100/\sqrt{d}$. Therefore, the existence of T and Z as above is impossible, completing the proof. □

4 Conclusions

We have presented an $O(n \log^4 n)$ time deterministic algorithm for the nuts and bolts matching problem. It is worth noting that since we applied expanders of polylogarithmic degrees there are simpler explicit constructions than those given in [10] and [11] and we can take appropriate Cayley graphs of the groups Z_2^k by applying some known constructions of Linear Error Correcting Codes, as described in [6]. This gives somewhat simpler graphs at the cost of increasing the complexity by a polylogarithmic factor. We omit the details.

It is conceivable that using the techniques of this paper one can get an $O(n \log^2 n)$ algorithm. However getting below $O(n \log^2 n)$ seems to require a new method. In particular our method does not reduce the set of active bolts at all, but keeps all of them "alive". Finding a $O(n \log n)$ algorithm seems to require a way of sampling from both the nuts and the bolts while keeping many of the sampled ones matched.

We can think of two problems where our deterministic sampling methods may be helpful: one is local sorting where the goal is to answer all the the relationships between elements who are neighbors in a given graph. An optimal probabilistic algorithm is known [9]. The other problem is selection where the input is stored in a read only memroy and ther is also some small read/write memory [12].

Acknowledgements

The last author wishes to thank Leonard Schulman for helpful discussions and Yishay Mansour for his hospitality while visiting Tel Aviv.

References

[1] M. Ajtai, J. Komlós, W.L. Steiger and E. Szemerédi, *Almost sorting in one round*, Advances in Computing Research, Vol. 5, 1989, JAI Press, pp. 117-126.

[2] N. Alon and Y. Azar, *Finding an approximate maximum*, SIAM J. on Computing 18, 1989, pp. 258-267.

[3] N. Alon and Y. Azar, *Parallel comparison algorithms for approximation problems*, Proc. 29^{th} IEEE Symp. on Foundations of Computer Science, Yorktown Heights, NY, 1988, pp. 194-203. Also: Combinatorica 11, 1991, pp. 97-122.

[4] N. Alon and F. R. K. Chung, *Explicit construction of linear sized tolerant networks*, Discrete Math. 72(1988), pp. 15-19; (Proc. of the First Japan Conference on Graph Theory and Applications, Hakone, Japan, 1986.)

[5] N. Alon and J. H. Spencer, **The Probabilistic Method**, Wiley, 1991.

[6] N. Alon and Y. Roichman, Random Cayley graphs and Expanders, Random Structures and Algorithms, in press.

[7] B. Bollobás and G. Brightwell, *Graphs whose every transitive orientation contains almost every relation*, Israel J. Math. 59, 1987, pp. 112-128.

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

[9] W. Goddard, C. Kenyon, V. King and L. Schulman, *Optimal randomized algorithms for local sorting and set-maxima*, SIAM J. on Comput. 22, 1993, pp. 272–283.

[10] A. Lubotzky, R. Phillips and P. Sarnak, *Explicit expanders and the Ramanujan conjectures*, Proc. 18^{th} ACM Symp. on Theory of Computing, 1986, pp. 240-246. See also: A. Lubotzky, R. Phillips and P. Sarnak, *Ramanujan graphs*, Combinatorica 8, 1988, pp. 261-277.

[11] G. A. Margulis, Explicit group-theoretical constructions of combinatorial schemes and their application to the design of expanders and superconcentrators, Problemy Peredachi Informatsii 24, 1988, pp. 51-60 (in Russian). English translation in Problems of Information Transmission 24 (1988), pp. 39-46.

[12] J. I. Munro and M. Paterson, *Selection and sorting with limited storage*, Theoretical Computer Science 12, 1980, pp. 315-323.

[13] N. Pippenger, *Sorting and selecting in rounds*, SIAM J. Comput. 6, 1987, pp. 1032-1038.

[14] G. J. E. Rawlins, **Compared to what? an introduction to the analysis of algorithms**, Computer Science Press, 1991.