Matching Nuts and Bolts Faster*

Phillip G. Bradford Rudolf Fleischer

Max-Planck-Institut für Informatik, Im Stadtwald, 66123 Saarbrücken, Germany.
E-mail: {bradford,rudolf}@mpi-sb.mpg.de.

Abstract. The problem of matching nuts and bolts is the following : Given a collection of n nuts of distinct sizes and n bolts such that there is a one-to-one correspondence between the nuts and the bolts, find for each nut its corresponding bolt. 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 very hard to solve. In fact, the best deterministic solution to date is due to Alon *et al.* [1] and takes $\Theta(n \log^4 n)$ time. Their solution uses (efficient) graph expanders. In this paper, we give a simpler $O(n \log^2 n)$ time algorithm which uses only a simple (and not so efficient) expander.

1 Introduction

In [7], 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 mathematician (instead of a carpenter) you would probably prefer to see the problem stated as follows ([1]) :

> *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 authors were 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 $\Omega(n \log n)$ 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 $O(n \log n)$, 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 [3]).

Unfortunately, it is much harder to find an efficient deterministic algorithm. The only one known to us is the algorithm by Alon *et al.* [1] which is also based on Quicksort. To find a good pivot element which splits the problem into two subproblems of nearly the same size, they run $\log n$ 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 a highly efficient expander of degree $\Theta(\log^2 n)$ to define its comparisons. Therefore, finding a good pivot takes $\Theta(n \log^3 n)$ time, and the entire Quicksort takes $\Theta(n \log^4 n)$ time.

In this paper, we propose a simpler algorithm to find a good pivot (see Section 3 for details). First, we connect the set of nuts with the set of bolts via some expander of constant degree and compare each nut to all the bolts to which it is connected by an edge of the expander. We discard all nuts which are only connected to smaller bolts or only connected to larger bolts. Then we play a simple knockout tournament on the remaining nuts (where in each round half of the nuts are eliminated) which guarantees that the winner of the tournament is a good pivot. Since we can play each round of the tournament in $O(n)$ time, we can find a good pivot in $O(n \log n)$ time. Therefore, we can solve the nuts and bolts matching problem in $O(n \log^2 n)$ time.

Alon *et al.* [1] mention two potential applications of this problem: the first is local sorting of nodes in a given graph [4], and the second is selection of read only memory with a little read/write memory [6].

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. And we conclude with some remarks in Section 4.

2 Basic Definitions

Let $S = \{s_1, \ldots, s_n\}$ be a set of nuts of different sizes and $B = \{b_1, \ldots, b_n\}$ 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 $rank_T(s) := \dfrac{|\{t \in T \mid s \geq t\}|}{|T|}$. If T is a multiset then the relative rank of s with respect to T is defined analogously, where each $t \in T$ is counted according to its multiplicity in 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 in $O(n \log n)$ time, where c is a small constant. Then our main result follows immediately.

Theorem 1. *We can match n nuts with their corresponding bolts in $O(n \log^2 n)$ time.*

Proof. The correctness of the algorithm above follows immediately from the correctness of Quicksort. For the running time observe, that each subproblem has size at most $(1-c)n$, hence the depth of the recursion is only $O(\log n)$, and in each level of the recursion we spend at most $O(n \log n)$ time to compute the c-approximate median and $O(n)$ time to split the problem into the two subproblems. $\square$

We now recall some facts about expanders (see for example [5] if you want to learn more about expanders). Let $0 < \alpha \leq \frac{1}{2}$ and $\alpha c < 1$. An (n, k, α, c)-*expander* is a k-regular bipartite graph on vertices I (inputs) and O (outputs), where $|I| = |O| = n$, such that every subset $A \subset I$ of size at most αn 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.

Theorem 2 (Alon, Galil, and Milman [2], Cor. 2.3). *If $n = m^2$ for some integer m, then we can construct an $(n, 9, \frac{1}{2}, 0.41)$-expander in $O(n)$ time.*

Corollary 3. *Let $0 < \delta \leq \frac{1}{2}$ and $\gamma_\delta = \frac{3-5\delta}{5\delta(1-\delta)}$. Then there exists an integer q_δ such that for any n, where $n = m^2$ for some integer m, we can construct an $(n, q_\delta, \delta, \gamma_\delta)$-expander in $O(n)$ time. In such an expander, any subset of the inputs of size δn is connected to at least $\frac{3}{5}n$ different outputs.*

Proof. We take a series of the expanders of Theorem 2 and identify the outputs O_1 of the first one with the inputs I_2 of the second one, the outputs O_2 of the second one with the inputs I_3 of the third one, and so on. Then there is an integer k_δ (independent of n) such that any set of δn inputs of I_1 is connected to at least

$\frac{n}{2}$ different outputs of O_{k_δ} and hence to at least $(1+\frac{0.41}{2})\frac{n}{2} > \frac{3}{5}n$ different outputs of $O_{k_\delta+1}$. We can easily calculate k_δ by computing the series defined by $a_0 := \delta$ and $a_{i+1} := a_i\,(1+0.41(1-a_i))$; then k_δ is the smallest index i such that $a_i \geq \frac{1}{2}$.

Hence, to get the desired bipartite graph, we only have to connect each node v of I_1 to all nodes w of $O_{k_\delta+1}$ which can be reached from v by traversing a path which uses exactly one edge from each of the $k_\delta+1$ expanders. Then the degree of any node is clearly at most $q_\delta := 9^{k_\delta+1}$. To make the graph q_δ-regular we can add arbitrary dummy edges without destroying the expansion property. Further, the expansion factor of the graph is at least γ_δ because $(1+\gamma_\delta(1-\delta))\,\delta n = \frac{3}{5}n$ (note that subsets of the inputs of size smaller than δn are even better expanded). $\square$

3 Finding a c-Approximate Median

Our algorithm to find a c-approximate median is based on a knockout tournament played on some subset of the nuts. We start with a subset $S_1 \subset S$ of the nuts where each nut $s \in S_1$ has a set $T_1(s)$ of two bolts associated with it; for all s, the sets $T_1(s)$ need not be disjoint, but every bolt may appear only in a constant number of them. We describe later how S_1 is constructed.

We then play $\lceil \log |S_1| \rceil$ rounds of the tournament, where in each round half the nuts survive for the next one. Intuitively, we take any two nuts together with their sets of associated bolts, determine which nut splits the union of both sets of bolts less equally, eliminate that nut, and give both sets of bolts to the surviving nut. Unfortunately, pairing the nuts arbitrarily does not quite work, i.e., the winner of the tournament would not necessarily be a c-approximate median, but there is a simple way to overcome that difficulty.

In general, let S_i be the set of nuts before we start round i. For each nut $s \in S_i$ let $T_i(s)$ be the multiset of bolts associated with s and let $r_i(s) := rank_{T_i(s)}(s)$ be the relative rank of s with respect to its set of bolts $T_i(s)$. Let $S_i^1 := \{s \in S_i \mid r_i(s) \geq \frac{1}{2}\}$ and $S_i^2 := \{s \in S_i \mid r_i(s) < \frac{1}{2}\}$.

We play the *knockout tournament* as follows.

$i := 1$;
while $|S_i| > 2$ **do**

(1) Pair the nuts of S_i^1 arbitrarily. If $|S_i^1|$ is odd then we eliminate the single nut without a partner.

(2) Let (s_1, s_2) be a pair of nuts from S_i^1. Compute the relative ranks of s_1 and s_2 with respect to the multiset $T := T_i(s_1) \cup T_i(s_2)$. Note that it is sufficient to compare s_1 with all bolts in $T_i(s_2)$ and s_2 with all bolts in $T_i(s_1)$, because $rank_T(s_j) = \frac{1}{2}(r_i(s_j) + rank_{T_i(s_{3-j})}(s_j))$, for $j = 1, 2$ (here we use Observation 4 (c)).
Whichever nut s has relative rank closer to $\frac{1}{2}$ survives in S_{i+1} and is associated with the multiset $T_{i+1}(s) := T$.

(3) Repeat steps (1) and (2) with S_i^2 instead of S_i^1.

od

Let l be the value of i after the **while**-loop terminates, i.e., $|S_l| \leq 2$. We claim that if S_1 was sufficiently large then every nut in S_l is a c-approximate median, where c is a small constant (see Lemma 5). But first we make a few simple observations.

Observation 4. *Assume we play the tournament starting with some set S_1 of nuts. Then*

(a) $\lceil \log |S_1| \rceil - 1 \leq l \leq \lceil \log |S_1| \rceil$.

(b) $S_l \neq \emptyset$.

(c) For $i = 1, \ldots, l$ and all $s \in S_i$, $|T_i(s)| = 2^i$. In particular, $|T_l(s)| \geq \frac{|S_1|}{2}$ for all $s \in S_l$.

(d) Each round needs $O(|S_1|)$ time.

Proof.

(a) In each round, we eliminate half of the nuts which could be paired, and at most two unpaired nuts, i.e., $|S_{i+1}| \geq \frac{|S_i|-2}{2}$. We stop if at most two nuts remain. It is easy to show by induction on $|S_1|$ that l must be at least $\log(|S_1|+2) - 1$. This proves the first inequality.
The second inequality follows directly from $|S_{i+1}| \leq \frac{|S_i|}{2}$.

(b) We never eliminate all nuts.

(c) By induction on i.

(d) Observe that in each round, every bolt is involved in at most one comparison (in step (2)). Since there are a total of $2|S_1|$ bolts in the first round and we never let additional bolts enter the game, we do at most $2|S_1|$ comparisons in each round. Further, pairing the nuts, computing the relative ranks, and merging two multisets of bolts do not increase the asymptotic complexity. $\square$

Lemma 5. *Let $S_1 \subseteq S$ and $\beta = \frac{|S_1|}{n}$. Suppose, each nut $s \in S_1$ lies between the two bolts in $T_1(s)$, i.e., $b_{\text{low}}(s) < s < b_{\text{high}}(s)$ if $T_1(s) = \{b_{\text{low}}(s), b_{\text{high}}(s)\}$, and every bolt appears at most q times in the sets $T_1(s)$. Then any $s \in S_l$ is a c-approximate median, where $c = \frac{\beta}{8q}$.*

Proof. Before the first round, we have $r_1(s) = \frac{1}{2}$ for all $s \in S_1$, and hence $\frac{1}{4} \leq r_2(s) \leq \frac{3}{4}$ for all $s \in S_2$. We now prove by induction, that this inequality holds after each round.

Assume we know that $\frac{1}{4} \leq r_i(s) \leq \frac{3}{4}$ for all $s \in S_i$. Let (s_1, s_2) be a pair from S_i^1, where w.l.o.g. $s_1 < s_2$. Let T be the multiset $T_i(s_1) \cup T_i(s_2)$. Since s_1 is larger than half of the bolts in $T_i(s_1)$, it must be larger than a quarter of the bolts in T. On the other hand, it is smaller than a quarter of the bolts in $T_i(s_1)$ and smaller than a quarter of the bolts in $T_i(s_2)$ (because it is smaller than s_2); hence it is smaller than a quarter of the bolts in T. Therefore, the inequality holds for s_1, and we only eliminate s_1 if the relative rank of s_2 with respect to T is even closer to $\frac{1}{2}$.

Let $s \in S_l$. Since $T_l(s)$ contains at least $\frac{\beta n}{2}$ bolts by Observation 4 (c), we conclude from the inequality above that s is larger than $\frac{\beta n}{8}$ bolts and smaller than another $\frac{\beta n}{8}$ bolts. Since each bolt may have up to q copies in $T_l(s)$, $\frac{\beta n}{8q} \leq rank(s) \leq (1 - \frac{\beta}{8q})n$, i.e., s is a $\frac{\beta}{8q}$-approximate median. □

Now we can give our algorithm to find a c-approximate median.

(1) If n is not the square of an integer, then add at most $2\sqrt{n}$ dummy nuts and bolts to make n the square of an integer.

(2) Let $B = I$ and $S = O$ be the sets of vertices of the $(n, q, \frac{1}{20}, \frac{220}{19})$-expander of Corollary 3. We compare each bolt with all the nuts to which it is connected by an edge of the expander.
Let S_1 be the set of nuts which are compared to at least one smaller bolt and at least one larger bolt. For all $s \in S_1$, pick arbitrarily one of the smaller and one of the larger bolts and put them into the set $T_1(s)$.

(3) Now play the knockout tournament starting with S_1. Let S_l be the set of (at most two) winners of the tournament. Choose any $s \in S_l$ as a good pivot.

Theorem 6. *This algorithm computes in $O(n \log n)$ time a c-approximate median, where $c = \frac{1}{80q}$ is a small constant not depending on n.*

Proof. Construction of the expander and hence of set S_1 takes $O(n)$ time (note that enlarging n slightly in step(1) does not increase the asymptotic complexity of the following steps). And the tournament takes $O(|S_1| \log |S_1|) = O(n \log n)$ time by Observation 4.

It remains to show that we really compute a c-approximate median for some constant c. First, observe that $|S_1| \geq \frac{n}{10}$. To see this, let B_1 be the set of bolts with rank at most $\frac{n}{20}$ and B_2 be the set of bolts with rank at least $\frac{19}{20}n$. Since the expander connects any subset of B of size $\frac{n}{20}$ to at least $\frac{3}{5}n$ different nuts in S, there must be a set of at least $\frac{n}{5}$ nuts which are connected to bolts in both B_1 and B_2. But then at least $\frac{n}{5} - 2 \cdot \frac{n}{20} = \frac{n}{10}$ of the nuts must have rank between $\frac{n}{20}$ and $\frac{19}{20}n$, which means they are in the set S_1.

Next, observe that each bolt appears in at most q of the sets $T_1(s)$ because the expander connects each bolt to exactly q nuts. Hence by Lemma 5, any $s \in S_l$ is a c-approximate median, where $c = \frac{1}{80q}$. □

4 Conclusions

We have presented an $O(n \log^2 n)$ time deterministic algorithm for matching nuts and bolts. This improves the first $O(n(\log n)^{O(1)})$-time solution of this problem, given by Alon *et al.*[1], by a factor of $\log^2 n$. As already mentioned in [1], the methods described in this (and their) paper seem not to be sufficient to reduce the complexity below $O(n \log^2 n)$.

Unfortunately, the constants hidden in our O-notation are incredibly large (far beyond the constant 10^8 in [1]). This is mainly due to the iterative construction in Corollary 3 which produces an expander of enormous, but still constant, degree (a short calculation shows that the degree is $q_{\frac{1}{20}} = 9^9$ because we must build the expander from 9 copies of the simple expander of Theorem2). On the other hand, the standard counting argument shows that there is a family of bipartite 24-regular graphs on $2n$ nodes which connect any subset of the inputs of size $\frac{n}{4}$ to at least $\frac{7}{8}n$ different outputs. Using these graphs in the construction of the set S_1 would give us an algorithm with fairly reasonable constants. However, we do not know an explicit construction for them. But it is clear that any improvement to our Corollary 3 can drastically reduce the running time of our algorithm (and make it more practical).

References

1. N. Alon, M. Blum, A. Fiat, S. Kannan, M. Naor and R. Ostrovsky. *Matching nuts and bolts.* Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'94), 1994, pp. 690–696.
2. N. Alon, Z. Galil and V.D. Milman. *Better expanders and superconcentrators.* Journal of Algorithms **8** (1987), pp. 337–347.
3. T.H. Cormen, C.E. Leiserson and R.L. Rivest. *Introduction to algorithms.* MIT Press, 1990.
4. W. Goddard, C. Kenyon, V. King and L. Schulman. *Optimal randomized algorithms for local sorting and set-maxima.* SIAM Journal on Computing 22, 1993, pp. 272–283.
5. A. Lubotzky. *Discrete groups, expanding graphs and invariant measures.* Birkhäuser Verlag, 1994.
6. J.I. Munro and M. Paterson. *Selection and sorting with limited storage.* Theoretical Computer Science 12, 1980, pp. 315–323.
7. G.J.E. Rawlins. *Compared to what ? An introduction to the analysis of algorithms.* Computer Science Press, 1992.