On Generalized Comparison-Based Sorting Problems

Jean Cardinal and Samuel Fiorini

Université libre de Bruxelles (ULB)
{jcardin, sfiorini}@ulb.ac.be

Abstract. We survey recent results on comparison-based sorting problems involving partial orders. In particular, we outline recent algorithms for partial order production and sorting under partial information. We emphasize the complementarity of the two problems and the common aspects of the algorithms. We also include open questions on two other related problems, namely partial order identification and sorting with forbidden comparisons.

1 Introduction

Sorting by comparison is a cornerstone of algorithms theory, and thorough analyses of comparison-based sorting and its relatives, such as linear-time median finding, multiple selection, and binary search trees, have been carried out since the seventies. Such analyses typically use powerful tools from analytic combinatorics.

However, optimal algorithms for simple generalizations of comparison-based sorting are not always known, or not well understood. The generalizations we consider in this paper tackle the following questions:

- What if only a partial, or approximate, ordering is required?
- What if some a priori knowledge is available on the ordering?
- What if the input data has no underlying total order?
- What if not all comparisons are allowed?

When properly formalized, these questions give rise to challenging theoretical problems. We will concentrate on versions of these problems involving partial orders and graphs.

In Section 2, we define two complementary sorting problems, called respectively *partial order production* and *sorting under partial information*. In Section 3 we define the entropy of a graph and explain its relevance to these problems. Finally, in Section 4 we describe algorithms for the two problems that are (nearly) optimal with respect to the number of comparisons they perform, and run in polynomial time. This first part of the paper summarizes the work done by a superset of the authors in collaboration with Ian Munro, and first published in 2009 and 2010. We chose to explain the two contributions in parallel, in order

to highlight the common features of the problems and their solutions. In fact, we believe that the two results are two facets of the same body of knowledge.

We consider another sorting problem, called *partial order identification*, in Section 5, and summarize known results about it. In contrast with the two previous ones, finding an efficient algorithm for this latter problem is still open, in a sense that we will make precise.

A fourth sorting problem, that we refer to as *sorting with forbidden comparisons*, is defined in Section 6. Although some special cases are well-studied, it seems that this question has received less attention than the ones above.

2 Sorting *to* and *from* a Partial Order

We now define our first two problems. In what follows, $e(P)$ denotes the number of linear extensions of a partially ordered set P.

2.1 Partial Order Production

In this problem, we are given a set P of n elements partially ordered by $\preceq$, and another set S of n elements with an underlying, unknown, total order $\leq$. We wish to find a bijection $f : P \mapsto S$, such that for every $x, y \in P$, we have $x \preceq y \Rightarrow f(x) \leq f(y)$. For this purpose, we are allowed to query the unknown total order $\leq$, and aim at minimizing the number of such queries.

Hence in this problem, the objective is to sort the data partially, by placing the items of S into bins, such that they obey the prescribed partial order relation $\preceq$ among the bins. Constructing a binary heap, for instance, amounts to placing items in nodes of a binary tree, so that the element assigned to a node is always smaller or equal to those assigned to the children of this node. The multiple selection problem can also be cast as the problem of partitioning the data into totally ordered bins of fixed sizes. Both problems are special cases of the partial order production problem.

The overall number of bijections is $n!$, but for the given partially ordered set P, we have $e(P)$ feasible bijections. Hence the information-theoretic worst-case lower bound on the number of comparisons for the partial order production problem is (logarithms are base 2):

$$\log n! - \log e(P).$$

Indeed, each query cuts out a part of the solution space that should contain, for the algorithm performing the sorting, half of the remaining bijections. One may stop querying when the remaining part of the solution space only contains feasible bijections. Because the solution space is initially of size $n!$ and the number of feasible bijections is $e(P)$, the number of queries one has to make is at least $\log n! - \log e(P)$ in the worst case.

The partial order production problem was first studied in 1976 by Schönhage [27], then successively by Aigner [1], Saks [26] and Bollobás and

Hell [3]. In his survey, Saks conjectured that the problem can be solved by performing a number of comparisons that is within a linear term of the lower bound in the worst case. Saks' conjecture was eventually proved in 1989 by Yao [30]. However, in the last section of his paper, Yao asked whether there exists an algorithm for the problem that is both query-optimal and runs in polynomial time.

2.2 Sorting with Partial Information

Here we are given a set P of n elements partially ordered by $\preceq$, and we seek an underlying, unknown, total order $\leq$ that extends $\preceq$. We wish to identify the total order by querying it, and aim at minimizing the number of such queries. Hence, this is the sorting problem in which the results of some comparisons are already known and given as input.

Since we wish to identify one of the $e(P)$ linear extensions of P, the information-theoretic lower bound on the worst-case number of comparisons for this problem is simply

$$\log e(P).$$

Note that the sum of the two lower bounds for the two problems is equal to $\log n!$, the lower bound for sorting. This is not surprising, since one can sort by first solving a partial order production problem for an arbitrary partial order, then solving an instance of sorting under partial information with the same partial order as input. In fact, this is exactly what *heapsort* does. In this algorithm, we first produce a partial order whose Hasse diagram is a binary tree, then sort the items in a total order using the partial information provided by the heap.

The problem of sorting under partial information was first posed by Fredman in 1976 [15], who showed that there exists an algorithm that performs $\log e(P) + 2n$ comparisons.

In 1984, Kahn and Saks [19] showed that there is a constant $\delta > 0$ such that every (non-totally ordered) poset has a query of the form "is $v_i < v_j$?" such that the fraction of linear extensions in which $v_i < v_j$ lies in the interval $[\delta, 1-\delta]$. They proved this for the constant $\delta = 3/11$. The well-known 1/3-2/3 conjecture (which remains open), formulated independently by Fredman, Linial and Stanley (see [22]) asserts that the same result holds for $\delta = 1/3$ (which, if true, is tight). Kahn and Linial [18] later gave a simpler proof of the existence of such a δ (with smaller value of δ). Brightwell, Felsner and Trotter [5], and Brightwell [4] further improved the value of δ.

Iteratively choosing such a comparison yields an algorithm that performs $O(\log e(P))$ comparisons. However, we do not know of any polynomial-time algorithm for finding a balanced pair, hence this does not lead to an algorithm with polynomial overall complexity. Indeed, computing the proportion of linear extensions that place v_i before v_j is $\sharp$P-complete [6].

In 1995, Kahn and Kim [17] described a polynomial-time algorithm performing $O(\log e(P))$ comparisons. Their key insight is to relate $\log e(P)$ to the entropy of the incomparability graph of P. We describe this notion below, and outline a simplification of their result yielding a more practical algorithm.

3 The Role of Graph Entropy

It is known that computing the number of linear extensions of a partial order is ♯P-complete [6]. Hence just computing the exact value of the lower bounds given above is out of reach. This is where graph entropy comes to the rescue. This notion will help us not only in finding a polynomial-time computable lower bound, but also in designing efficient algorithms for both problems.

We will only define graph entropy for comparability graphs here, which is exactly what we need for tackling our sorting problems. Consider a set P of n elements partially ordered by $\preceq$. To each element v we associate an open interval $I(v) \subseteq (0,1)$ in such a way that whenever $v \preceq w$, interval $I(v)$ is entirely to the left of interval $I(w)$. We obtain a collection $\{I(v)\}_{v \in P}$ of intervals that is said to be *consistent* with $\preceq$. The *entropy* of P is then defined as

$$H(P) := \min_{\{I(v)\}_{v \in P}} \frac{1}{n} \sum_{v \in P} \log \frac{1}{\text{length}(I(v))},$$

where the minimum is taken over all collections of intervals consistent with $\preceq$.

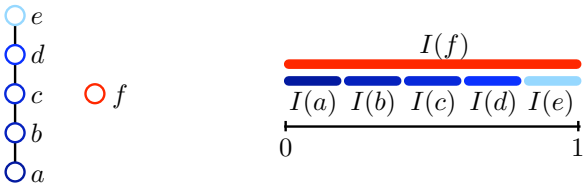

Fig. 1. A partially ordered set and its optimal consistent collection of intervals.

The entropy of P in fact only depends on its comparability graph G, and coincides with the *graph entropy* $H(G)$, as first defined by Körner [21]. Because G is a perfect graph, the graph entropy of its complement $\bar{G}$ satisfies $H(\bar{G}) = \log n - H(G)$ (see Csiszár et al. [9]). This leads to defining

$$H(\bar{P}) := \log n - H(P),$$

which is exactly the graph entropy of the incomparability graph $\bar{G}$ of P.

This definition of $H(P)$ was instrumental in our work [7, 8], although graph entropy admits several definitions and appears in other contexts [28].

Intuitively, $H(P)$ measures the average amount of information per element of P, and the optimal consistent collection of intervals tries to capture what the average linear extension of P looks like. Think of taking a random point in each interval; what we get as a result is a linear extension of P. Clearly, small intervals give us more information about the resulting linear extension than big intervals. For instance, in Figure 1, the contribution of f to the total amount of information is $\log 1 = 0$ and that of a is $\log 5 \approx 2.32$.

In case P is an antichain, $H(P) = 0$. In case P is a chain, $H(P) = \log n$. Between these two extreme cases, $H(P)$ is monotonic in the sense that adding comparabilities to P can never decrease $H(P)$. In particular, we always have $0 \leq H(P) \leq \log n$. For example, when P is formed of a chain of $n-1$ elements plus one element that is incomparable to all others, $H(P) = \frac{n-1}{n} \log(n-1) = \log n - \frac{1}{n} \log n - \Theta\left(\frac{1}{n}\right)$, see Figure 1 for an illustration for $n = 6$.

Similarly, $H(\bar{P})$ measures the average amount of uncertainty per element of P. For instance $H(\bar{P}) = 0$ in case P is a chain, $H(\bar{P}) = \log n$ in case P is an antichain. We have $0 \leq H(\bar{P}) \leq \log n$ and adding comparabilities to P can never increase $H(\bar{P})$.

The quantities that we will use are $nH(P)$ and $nH(\bar{P})$. These quantities measure the total amount of information and uncertainty in P, respectively. They yield lower bounds for the partial order production problem and the problem of sorting under partial information, respectively. The intuition is that each query produces information or equivalently, reduces the uncertainty, about the underlying unknown linear order. For producing P, we have to create at least as much information as contained in P. For sorting with partial information P, we have to reduce the uncertainty from that inherent in P to 0.

Theorem 1. *Let P be a partially ordered set of n elements. Then,*

$$nH(P) \leq \log n! - \log e(P) + \log e \cdot n, \tag{1}$$

$$nH(\bar{P}) \leq 2 \log e(P). \tag{2}$$

Eq. (1) is based on a simple volume argument using the so-called order polytope, see [7]. Kahn and Kim [17] proved that $nH(\bar{P}) \leq c \log e(P)$ with a constant c slightly larger than 11. Eq.(2) was proved in [8], and is tight (take P to be an antichain of two elements).

Now, we sketch a simple proof of the weaker inequality $nH(\bar{P}) \leq 4 \log e(P)$ that crucially relies on our interval-based definition of the entropy of partially ordered sets. The reader is referred to [8] for more details. The whole argument boils down to the following game: a player picks two incomparable elements a and b in P and gives them to an oracle. The oracle decides which of the two comparabilities $a \prec b$ or $b \prec a$ should be added to P. The goal for the player is to pick a and b in such a way that the entropy of the resulting partially ordered set P' always satisfies $nH(P') \leq nH(P) + 4$.

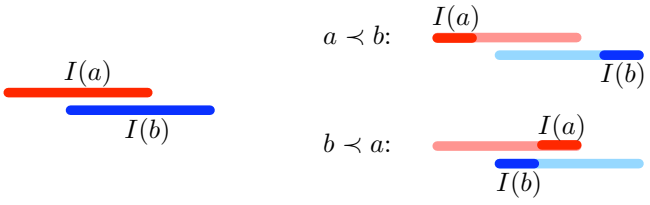

Fig. 2. The proof idea: take quarters of intervals.

A winning strategy for the player is to compute the optimal consistent collection of intervals $\{I(v)\}_{v \in P}$ for P and then pick a and b such that the length of $I(a)$ is maximum and $I(b)$ contains the midpoint of $I(a)$. If the oracle answers $a \prec b$, the player changes the collection of intervals by replacing $I(a)$ by its first quarter and $I(b)$ by its last quarter. Otherwise, the oracle answers $b \prec a$ and the players replaces $I(a)$ by its last quarter and $I(b)$ by its first quarter. This is illustrated in Fig 2. In both cases, the player's changes to the collection of intervals causes an increase of $2 \log 4 = 4$ in the sum

$$\sum_v \log \frac{1}{\text{length}(I(v))}.$$

Hence, $nH(P') \leq nH(P) + 4$ in both cases.

4 Approximating the Entropy and Efficient Algorithms

At the heart of our method lies a greedy algorithm for coloring the comparability or incomparability graph of a partially ordered set P. This algorithm simply iteratively finds a maximum size independent set (or stable set), makes it a new color class and removes it from the graph, until no vertex is left. When the greedy coloring algorithm is applied to the comparability graph of P, we obtain a greedy antichain decomposition of P. When it is applied to the incomparability graph of P, we obtain a greedy chain decomposition of P. This is illustrated in Figure 3.

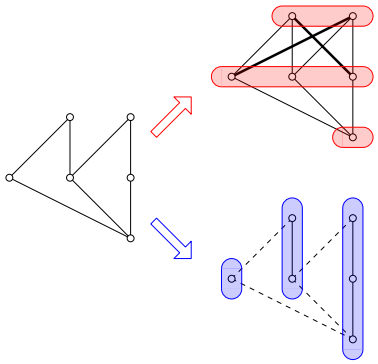

Fig. 3. Greedy antichain and chain decompositions.

Any coloring of G has an *entropy*, defined as the Shannon entropy of the size distribution of its color classes. If k is the number of color classes and p_i denotes the fraction of vertices in the ith color class, then the entropy of the coloring is $\sum_{i=1}^{k} p_i \log \frac{1}{p_i}$. This actually equals the graph entropy of the graph obtained

from G by adding all edges between any two color classes. The key fact is that the entropy of this new graph is not much bigger than that of G, provided that G is perfect.

Theorem 2 ([7]). *Let G be a perfect graph on n vertices and denote by g the entropy of a greedy coloring of G. Then*

$$g \leq H(G) + \log H(G) + O(1). \tag{3}$$

We now explain how to design algorithms for the two problems that rely on greedy colorings. For the sake of a concise and readable exposition, we deliberately skip many important details.

The main idea is to replace the target or partial information P by a partially ordered set Q with a simpler structure, such that solving the problem with P replaced by Q also solves the problem for P. This may force the algorithm to perform more comparisons, but not much more, thanks to Theorem 2. Thus we trade a few extra comparisons for a major gain in terms of structure. The transformations for the two problems are illustrated on Figure 3.

4.1 From Partial Order Production to Multiple Selection

For the partial order production problem, we add comparabilities to P to transform it into a layered partially ordered set, where the elements in each layer are mutually incomparable. This is achieved by a greedy antichain decomposition of the order, as illustrated on top of Figure 3. Now the entropy of the layered order is equal to that of the greedy coloring of the comparability graph and Theorem 2 applies. (Note that it is well possible that applying the greedy coloring algorithm blindly can yield a collection of antichains that is not totally ordered. We can solve this issue by applying an uncrossing step that preserves the value of the entropy, or by applying the greedy coloring algorithm twice. Details are given in the original paper [7].)

When P is a layered partial ordered set, the partial order production problem is essentially equivalent to the multiple selection problem, for which Kaligosi, Mehlhorn, Munro and Sanders [19] give an algorithm that requires only at most $nH(P)+o(nH(P))+O(n)$ comparisons. Combining this with Theorems 1 and 2, we obtain a polynomial-time algorithm for the partial order production problem performing at most $\log n! - \log e(P) + o(\log n! - \log e(P)) + O(n)$ comparisons. Since there exists a simple $\Omega(n)$ adversarial lower bound for the problem, provided that the comparability graph of P is connected, our algorithm is query-optimal.

4.2 From Sorting under Partial Information to Multiple Merging

For the problem of sorting under partial information, we remove comparabilities from P and turn it into a union of chains. We then sort this collection of chains by iteratively picking the two minimum size chains and merging them, following

a known strategy proposed by Frazer and Bennett [14]. This exactly mimics the construction of a Huffman tree, and therefore the query complexity is easily seen to be related to the entropy of the coloring. Combining again the bounds of Theorems 1 and 2, this leads to a $O(n^{2.5})$ time algorithm performing at most $(1+\varepsilon)\log e(P) + O_\varepsilon(n)$ comparisons.

Clearly, the information-theoretic lower bound can be sublinear for this problem. For instance, think of a chain with $n-1$ elements with 1 extra element incomparable to all others as in Figure 1. There, the lower bound is $\log e(P) = \log n$ and $nH(\bar{P}) = \log n + \Theta(1)$. To obtain a query-optimal algorithm, extra work is needed. Putting aside the largest chain in the greedy chain decomposition and carefully merging this chain with the rest in a last step, remembering the comparabilities between the two parts, leads to a query-optimal algorithm. The interested reader is again referred to the original paper [8].

For both of the problems, we have query-optimal algorithms whose work can be divided in two phases: a first costly phase where P is carefully analyzed, that takes $O(n^{2.5})$ time but where no comparison is performed; a second phase where comparisons are done that takes $O(B)+O(n)$ time, where B is the corresponding information theory lower bound [7, 8].

5 Partial Order Identification

We now consider the problem of identifying a partial order. Given a set P of n elements, with an unknown, underlying partial order $\preceq$, we wish to identify this partial order by querying it, and aim at minimizing the number of such queries. We suppose that we have access to an oracle that, given two elements a and b, tells us that either $a \preceq b$, $b \preceq a$, or a and b are incomparable. An algorithm for a given input size n can therefore be modeled as a ternary decision tree.

Partial order identification was studied in particular by Faigle and Turán in 1988 [13]. In this contribution, they present a number of other related problems, and a number of algorithms for the identification problem. The problem was also tackled by Dubhashi et al. [11], and much more recently by Daskalakis et al. [10].

The problem has a trivial $\Omega(n^2)$ lower bound in terms of number of comparisons, as $\binom{n}{2}$ comparisons are necessary to identify an empty order, in which all pairs are incomparable. In order to provide more interesting lower bounds, we introduce two parameters associated with a partial order P. The first parameter is the number N of downsets of P, where an downset is a subset of P that is closed for the relation $\preceq$. Hence D is an downset if and only for every $x \in D$, all elements $y \preceq x$ are also in D. The second parameter is w, the width of the order, that is, the size of a largest antichain in P. Note that while a largest antichain can be found in polynomial time, counting the number of downsets is again ♯P-complete [24].

5.1 Partial Order Identification using Central Elements

The following lower bound was proved by Dubhashi et al. [11]:

Theorem 3. *The worst-case number of comparisons for solving the poset identification problem, given that the poset $(P, \preceq)$ is guaranteed to have at most N downsets, is $\Omega(n \log N)$ (where $n = |P|$).*

This lower bound comes from the fact that the number of partial orders with n elements and at most N downsets is exponential in $n \log N$. It appears to be achievable via a generalization of insertion sort: insert the elements one at a time in the poset induced by the previous elements. This can be carried out using a number of queries proportional to $\log N$, thanks to the existence of a so-called *central element*, a classical result due to Linial and Saks [23].

Theorem 4. *In every poset, there exists an element such that the fraction of downsets containing this element is between δ and $1 - \delta$, for a certain universal constant δ (at least 0.17).*

This algorithm was proposed by Faigle and Turán [13], although they do not seem to have noticed the optimality of the algorithm. For each element x, it performs two binary searches, one for identifying the downset consisting of elements smaller than x, and another one for the elements greater than x. The remaining elements must be those that are incomparable. The binary searches use the central elements as pivots. Here is a more detailed outline of the algorithm that, given a poset $(P, \preceq)$ and a new element $x \notin P$, identifies the downset of elements of P smaller than x. We use the notations $D_P(y) := \{z \in P : z \preceq y\}$ and $U_P(y) := \{z \in P : z \succeq y\}$.

1. **Bisect** (x, P)
2. if $P = \emptyset$, return $\emptyset$
3. choose a central element $y \in P$
4. if $x \succ y$:
   (a) let $P' := P \setminus D_P(y)$
   (b) let $Q :=$ Bisect (x, P')
   (c) return $D_P(y) \cup Q$
5. else:
   (a) let $P' := P \setminus U_P(y)$
   (b) return Bisect (x, P')

It can be checked that the number of remaining downsets after each comparison is cut down by a constant factor. Indeed, in the case where $x \succ y$, there cannot be more downsets in $P \setminus D_P(y)$ than downsets in P that contain y. Otherwise, all the downsets in $P \setminus U_P(y)$ are in one-to-one correspondence with downsets of P that do not contain y. In both cases, this number is a constant fraction of N.

However, the bisection, and therefore the whole insertion sort algorithm, is not known to admit a polynomial-time implementation. This is because we are so far not able to identify a central element in polynomial time. Indeed, any polynomial-time probabilistic algorithm for finding a δ-central element of a poset would yield a polynomial-time algorithm for estimating its number of downsets to within an arbitrary small factor with high probability [11].

Finding a 1/4-central element can be done in polynomial time in a 2-dimensional poset, as shown by Steiner [29]. Faigle et al. [12] study other special cases of posets, and give polynomial-time algorithms for finding central elements in interval and series-parallel orders. Hence identifying 2-dimensional, interval, and series-parallel orders can be done optimally in terms of queries, and the algorithm is polynomial.

5.2 Partial Order Identification using Chain Decompositions

An alternative algorithm was proposed by Faigle and Turán [13] to implement the insertion sort procedure. For each inserted element x, this algorithm computes a chain decomposition of the current poset, and performs a bisection search for x in each of the chains. This algorithm has query complexity $O(nw\log(n/w))$, where w is the width of the poset. This is straightforward from the fact that there exists a chain decomposition into w chains (Dilworth's Theorem), which can be found in polynomial time. However, this is not tight with respect to the following lower bound as a function of w, proved by Daskalakis et al. [10].

Theorem 5. *The worst-case number of comparisons for solving the poset identification problem, given that the poset $(P, \preceq)$ is guaranteed to have width at most w, is $\Omega(n(\log n + w))$.*

Note that the lower bound with respect to w is not comparable to the lower bound with respect to N, since the sets of possible inputs are distinct. Daskalakis et al. [10] also give an algorithm that reaches the $\Omega(n(\log n + w))$ lower bound. However, again, this algorithm is not polynomial. In particular, it requires the exact computation of the number of linear extensions (under some constraints) of the current poset. They also propose a practical implementation of Faigle and Turán's suboptimal chain decomposition strategy.

Hence, to the best of our knowledge, the following question is still open: find a query-optimal polynomial-time algorithm for partial order identification, where query-optimality is with respect to either the width w or the number of downsets N.

6 Sorting with Forbidden Comparisons

In this last section, we briefly summarize previous results on sorting problems involving *forbidden comparisons*, that is, in which some designated pairs of elements cannot be compared.

A well-known special case of this problem is the so-called *nuts and bolts* problem. Let $B = \{b_1, b_2, \ldots, b_n\}$ (the bolts) be a set of totally ordered elements (say, real numbers), and let $S = \{s_1, s_2, \ldots, s_n\}$ (the nuts) be a permutation of B. The numbers correspond to the widths of the nuts and bolts, and the goal is to match them. The only comparison operations that are allowed are between nuts and bolts, that is, we are only allowed to test whether $s_i \leq b_j$ for some pair $i, j \in [n]$. It is not difficult to find a randomized Quicksort-like

algorithm that runs in expected $O(n \log n)$ time [25]. In 1994, Alon et al. [2] proposed a deterministic algorithm running in time $O(n(\log n)^{O(1)})$. Komlós, Ma and Szemerédi gave a deterministic $O(n \log n)$ algorithm in 1996 [20].

The problem of sorting with forbidden comparisons generalizes the nuts and bolts problem. We are given a graph $G = (V, E)$. The set V is the set of elements to sort, and E is the set of pairs for which comparison is allowed. When we compare a pair $(u, v) \in E$, we are given the orientation of the edge uv in G corresponding to the order of the elements u, v. The goal is to unveil the underlying total order on V. We are also promised that probing all edges effectively yields a total order. Hence G has a Hamiltonian path, and the orientation of the edges that are not in this path are implied by transitivity.

In 2011, Huang et al. [16] proposed an algorithm performing $O(n^{3/2} \log n)$ queries. This algorithm maintains a likeliness information about the orientation of each edge, and implements a two-way strategy that at each step identifies an edge to probe, or finds the orientation of the edges incident to a subset of $O(\sqrt{n})$ vertices in $O(n \log n)$ time. A subroutine of the algorithm involves estimating the average ranks of the elements by sampling the order polytope, a technique reminiscent of the ones used for sorting under partial information. There is not any interesting lower bound for this problem, however, apart from the standard $\Omega(n \log n)$ bound for sorting. Increasing this lower bound would definitely be an important progress.

Acknowledgments. This work is supported by the ARC Convention AUWB-2012-12/17-ULB2 (COPHYMA project), and the ESF EUROCORES programme EuroGIGA, CRP ComPoSe, F.R.S.-FNRS Grant R70.01.11F. We thank our coauthors on [7, 8], Gwenaël Joret for proofreading a draft of this manuscript, as well as the anonymous referees for their careful reading and many important precisions.

References

1. M. Aigner. Producing posets. *Discrete Math.*, 35:1–15, 1981.
2. N. Alon, M. Blum, A. Fiat, S. Kannan, M. Naor, and R. Ostrovsky. Matching nuts and bolts. In *SODA*, pages 690–696, 1994.
3. B. Bollobás and P. Hell. Sorting and graphs. In *Graphs and order, Banff, Alta., 1984*, volume 147 of *NATO Adv. Sci. Inst. Ser. C Math. Phys. Sci.*, pages 169–184, Dordrecht, 1985. Reidel.
4. G. R. Brightwell. Balanced pairs in partial orders. *Discrete Mathematics*, 201(1–3):25–52, 1999.
5. G. R. Brightwell, S. Felsner, and W. T. Trotter. Balancing pairs and the cross product conjecture. *Order*, 2(4):327–349, 1995.
6. G. R. Brightwell and P. Winkler. Counting linear extensions. *Order*, 8(3):225–242, 1991.
7. J. Cardinal, S. Fiorini, G. Joret, R. M. Jungers, and J. I. Munro. An efficient algorithm for partial order production. *SIAM J. Comput.*, 39(7):2927–2940, 2010.
8. J. Cardinal, S. Fiorini, G. Joret, R. M. Jungers, and J. I. Munro. Sorting under partial information (without the ellipsoid algorithm). In *STOC*, pages 359–368, 2010. To appear in *Combinatorica*.

9. I. Csiszár, J. Körner, L. Lovász, K. Marton, and G. Simonyi. Entropy splitting for antiblocking corners and perfect graphs. *Combinatorica*, 10(1):27–40, 1990.
10. C. Daskalakis, R. M. Karp, E. Mossel, S. Riesenfeld, and E. Verbin. Sorting and selection in posets. *SIAM J. Comput.*, 40(3):597–622, 2011.
11. D. P. Dubhashi, K. Mehlhorn, D. Ranjan, and C. Thiel. Searching, sorting and randomised algorithms for central elements and ideal counting in posets. In *FSTTCS*, pages 436–443, 1993.
12. U. Faigle, L. Lovász, R. Schrader, and G. Turán. Searching in trees, series-parallel and interval orders. *SIAM J. Comput.*, 15(4):1075–1084, 1986.
13. U. Faigle and G. Turán. Sorting and recognition problems for ordered sets. *SIAM J. Comput.*, 17(1):100–113, 1988.
14. W. D. Frazer and B. T. Bennett. Bounds on optimal merge performance, and a strategy for optimality. *J. ACM*, 19(4):641–648, 1972.
15. M. L. Fredman. How good is the information theory bound in sorting? *Theor. Comput. Sci.*, 1(4):355–361, 1976.
16. Z. Huang, S. Kannan, and S. Khanna. Algorithms for the generalized sorting problem. In *FOCS*, pages 738–747, 2011.
17. J. Kahn and J. H. Kim. Entropy and sorting. *J. Comput. Syst. Sci.*, 51(3):390–399, 1995.
18. J. Kahn and N. Linial. Balancing extensions via Brunn-Minkowski. *Combinatorica*, 11:363–368, 1991.
19. K. Kaligosi, K. Mehlhorn, J. I. Munro, and P. Sanders. Towards optimal multiple selection. In *ICALP*, pages 103–114, 2005.
20. J. Komlós, Y. Ma, and E. Szemerédi. Matching nuts and bolts in o(n log n) time. *SIAM J. Discrete Math.*, 11(3):347–372, 1998.
21. J. Körner. Coding of an information source having ambiguous alphabet and the entropy of graphs. In *Transactions of the 6th Prague Conference on Information Theory*, pages 411–425, 1973.
22. N. Linial. The information-theoretic bound is good for merging. *SIAM J. Comput.*, 13(4):795–801, 1984.
23. N. Linial and M. Saks. Every poset has a central element. *J. Comb. Theory, Ser. A*, 40(2):195–210, 1985.
24. J. S. Provan and M. O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM J. Comput.*, 12(4):777–788, 1983.
25. G. J. E. Rawlins. *Compared to what? - an introduction to the analysis of algorithms*. Principles of computer science series. Computer Science Press, 1992.
26. M. E. Saks. The information theoretic bound for problems on ordered sets and graphs. In *Graphs and order, Banff, Alta., 1984*, volume 147 of *NATO Adv. Sci. Inst. Ser. C Math. Phys. Sci.*, pages 137–168, Dordrecht, 1985. Reidel.
27. A. Schönhage. The production of partial orders. In *Journées algorithmiques, École Norm. Sup., Paris, 1975*, pages 229–246. Astérisque, No. 38–39. Soc. Math. France, Paris, 1976.
28. G. Simonyi. Graph entropy: a survey. In *Combinatorial optimization (New Brunswick, NJ, 1992–1993)*, volume 20 of *DIMACS Ser. Discrete Math. Theoret. Comput. Sci.*, pages 399–441. Amer. Math. Soc., Providence, RI, 1995.
29. G. Steiner. Searching in 2-dimensional partial orders. *J. Algorithms*, 8(1):95–105, 1987.
30. A. C. Yao. On the complexity of partial order productions. *SIAM J. Comput.*, 18(4):679–689, 1989.