Zeitschrift: L'Enseignement Mathématique
Herausgeber: Commission Internationale de l'Enseignement Mathématique
Band: 28 (1982)
Heft: 1-2: L'ENSEIGNEMENT MATHÉMATIQUE

Artikel: STRUCTURED vs GENERAL MODELS IN COMPUTATIONAL COMPLEXITY
Autor: Borodin, A.
Kapitel: II. Comparison Based Models
DOI: https://doi.org/10.5169/seals-52236

I would argue that these barriers and many of these same conjectures are of fundamental importance in the structured setting although here one has to look at each specific model to formulate appropriate questions. But this is precisely my main, albeit obvious, conclusion—namely, that it is important and productive to formulate and study the analogous barriers and conjectures in reasonably natural structured settings. Of course, I should admit that my perspective may distort the fact that specific instances of these questions were studied in structured settings *before* the issues were formulated generally. But in the general framework these issues have come into clearer focus. There are several reasons why these issues should also be pursued in structured settings.

1. The issues are usually of significant independent interest in the different settings, especially when the model represents the "natural" model.
2. Some structured models have the property, often by design, that they are sufficiently "general with respect to a specific complexity issue" that results in such a setting will yield a direct corollary for the general theory.
3. A structured model may not be sufficiently general to yield direct corollaries but nevertheless the proof techniques which are developed may become paradigms for the general model.

In the following sections I would like to substantiate these points by primarily considering the two structured models, S1 (comparison based models) and S2 (arithmetic models) mentioned initially. We will then discuss a few other examples outside of these models to further emphasize the utility of this viewpoint.

II. Comparison Based Models

I want to concentrate on a few examples of models which hopefully will exemplify the utility of the structured viewpoint. The first model, or rather class of models, is the comparison tree (see Knuth [73]). In a pure comparison tree, we label the nodes of a tree by questions of the form "$x_i \leqslant x_j$?". The model then can only solve problems dealing with "searching and sorting". It is also sufficient to consider the input domain to be $\{1, 2, \ldots, n\}$ for a problem of size n. On a given input, the computation follows an appropriate path to a leaf, where the output takes place. Since

every problem under consideration is completely determined by the permutation of the input, and since we can sort in $n \log n + o(n)$ comparisons, this simple model cannot address itself to many of the "larger issues" (e.g. P vs NP). Yet, we do get an $\Omega(n \log n)$ lower bound not only for sorting but for set recognition problems like "X distinct ?", "$X = Y$?". The sorting argument simply observes that we need at least one leaf for each of the possible $n!$ permutations and hence the depth of the tree (= number of comparisons = Time in this model) must be at least $\lceil \log n! \rceil = n \log n + o(n)$. As simple as this argument is, it provides a paradigm for asking and answering the same complexity question in a more interesting setting, namely for Random Access Machines (see Paul [80]) with $+$, $-$, $\times$ as unit cost operations. The same questions apparently remain open if integer division is also allowed as a unit cost operation.

One way to establish the set recognition lower bounds can be obtained by considering an extension of the model, namely linear comparison trees where nodes are labelled "$\sum_{i=1}^{n} c_i x_i \geqslant c_{n+1}$?", with $\{c_i\}$ in some underlying field (say $\mathbf{Q}$ for definiteness). For linear comparison trees, we can consider the input domain to be $\mathbf{Q}^n$. It is easily seen that the set of inputs leading to any leaf is a convex subset of $\mathbf{Q}^n$. Dobkin and Lipton [78] then observe that if a subset A (of $\mathbf{Q}^n$) which is being recognized is the disjoint union of k open sets, we must have at least one leaf (in the linear comparison tree) for each such open set (by convexity). Again, it follows that $\lceil \log k \rceil$ time is required. For example, if $A = \{ < x_1, ..., x_n > \mid x_i \neq x_j \}$ it follows that $k = \Omega(n!)$ and hence the $\Omega(n \log n)$ lower bound. By the same proof technique, Dobkin and Lipton show that the knapsack problem ($\{ < x_1, ..., x_n > \mid \exists i_1, ..., i_r : \sum x_{i_j} = 1 \}$) requires $\Omega(n^2)$ comparisons.

The linear tree model allows us to pose more challenging questions; that is, beyond what can be done with a pure comparison tree. Although we can view linear trees as an extension of the pure model, I would claim that, relative to its scope of intended problems, this model is in a sense more structured. This will become clearer if we enlarge our discussion to Space considerations (where it is possible to force larger Time bounds). Each node of a comparison tree can be thought of as representing a state (or I.D.) of the computation. In order to introduce Space complexity, we should coalesce identical states (that is, those with identical subtrees) and let outputs take place at any step of the computation. We are then led to Pippenger's comparison branching programs (see Tompa [78]), which are directed acyclic graphs (rather than trees) whose nodes are labelled as in

comparison trees and whose edges are labelled to denote possible outputs. The Space used by a branching program is defined as log (# nodes or states), which is precisely Cobham's [66] notion of *capacity* which was defined for general models. (We should also note that a general version of branching programs was also studied by Masek [76] prior to their introduction into a structured setting).

We can construct branching programs for any set of predicates, in particular we can have $\{=, \neq\}$, $\{\leqslant, >\}$, or linear comparisons. And now we can try to clarify why linear branching programs appear to be more structured. Cook, and Tompa [78] observe that $\{\leqslant, >\}$ (or $\{=, \neq\}$) branching programs are "general with respect to Space and Time (within a factor of n) complexity" in the following sense:
Suppose we have a problem for which we can establish that any branching program which works correctly for problem size n must have Space $= \Omega(S(n))$ (or for which we can establish a Time-Space tradeoff of the form f (Time, Space) $= \Omega(P(n))$—e.g. Space $= O(\log^k n) \Rightarrow$ Time $= \Omega(n^{\log n})$). Then a corresponding result will hold for a general model because the structured model can simulate the general model on a "representative set of inputs". Suppose the $\{\leqslant, >\}$ (respectively, $\{=, \neq\}$) comparison problem is to compute $f_1(x_1, ..., x_n), ..., f_r(x_1, ..., x_n)$; the analogous general problem is to output $\overline{f_1(x, ..., x_n)}, ..., \overline{f_r(x_1, ..., x_n)}$ given $\bar{x}_1, ..., \bar{x}_n$ where $\bar{y}$ denotes a binary encoding of the integer y. (Note, the model insures that each $f_j(x_1, ..., x_n) = x_{i_j}$ for some index i_j). We only assume that the general model has a read-only input (not necessarily a tape), with a fixed number of reading heads. In fact, since we are permitting random access on the input we can assume that there is only one input head. Now suppose the general model solves this analogous general problem in Space (= Capacity) $\tilde{S}(n)$, and Time (= # of steps or just read instructions) $\tilde{T}(n)$. In particular, the general machine must work on a special class of inputs, those that satisfy the property that x_i is the rank of x_i in $\{x_1, ..., x_n\}$ (resp. x_i is the smallest index $j(i) \leqslant i$ such that $x_i = x_{j(i)}$). But now the structured comparison program can simulate the behaviour of the general machine on this class of inputs in the following way: we need to simulate a move δ: (present state, input bit being read) $\mapsto$ (new state, new head position). But we can use the $\{\leqslant, >\}$ (resp. $\{=, \neq\}$) tests to determine any x_i (and hence any bit of x_i) using at most $n-1$ (resp. $i-1$) comparisons so that the branching program has Time complexity

$T \leqslant n\tilde{T}$, and increasing the number of states by a factor of n^2 (resp. n) so that the Space S of the branching program is $O(\tilde{S})$ since Space $\geqslant \log n$ (whenever the output depends on all n inputs). The bounds S and T then apply as upper bounds within the structured model for the original problem.

This is the sense in which we mean that such a model is "general relative to a particular issue". The model is not general (in that it cannot always get at the encoding), but it can simulate the general machine on a "representative set of inputs". Savitch [73], and Cook and Rackoff [80] have made such observations before with respect to conjecture GC2 (see section IV).

The question then arises as to whether or not linear tree or linear branching programs may be sufficiently general in this same sense. But now it is not clear whether or not there exists an appropriate representative set of inputs for the type of problems we would like to consider. I want to mention one such problem, the shortest path problem, relating to conjecture GC5. The problem can be formulated as a set recognition problem or as a function; given $A = (a_{ij})$, where a_{ij} is the distance (or cost) associated with edge $< i, j >$, compute $D = (d_{ij})$, where d_{ij} is the distance (or the path itself) of the shortest (i.e. of least cumulative cost) path from node i to node j. This problem has received considerable attention from the point of view of Time complexity. The most structured model for the problem is a straight-line program or circuit (i.e. no predicates) with operations "min" and $+$. Kerr [70] showed that such "oblivious (the sequence of operations is independent of the inputs) programs require $\Omega(n^3)$, where n is the number of nodes and hence n^2 is the size of the problem. A more challenging setting is provided by linear tree programs. In this setting Fredman [76] demonstrates an $O(n^{2.5})$ method (which can be used as the basis for a $O(n^3 (\log\log n)^{1/3}/(\log n)^{1/3}) = o(n^3)$ uniform algorithm). With regard to lower bounds, we only have a negative result by Rivest and Yao [78] that a particularly appealing approach cannot yield a sought after $\Omega(n^2 \log n)$ lower bound. I am interested in Time-Space bounds for this problem in the context of linear branching programs. It seems necessary to extend the model to allow assignments $y_i :=$ linear combination of previously defined $\{y_j\}$ and inputs $\{x_k\}$. Then Space is defined as Capacity plus the number of extra variables y_i. The shortest path problem is an excellent example of a problem which is in $P \cap \bigcup_k \text{DSPACE}(\log^k)$ but not, apparently, in Polytimelogspace. (Here I use these terms to have the obvious meaning for a structured setting like linear branching programs as well as their more standard meaning in the general setting). It is con-

ceivable to me that ideas from linear geometry may enable someone (apparently, not me) to establish an $\omega(\log n)$ lower bound on Space, and even more generally establish the structured analogue of conjecture GC5. But I don't see how this would directly yield a corollary for the general theory. It seems fair to augment the Space measure by the precision of the coefficients used in the program to reflect the fact that such coefficients would have to be represented. Now given bounds on Space and Time, we can put bounds on the precision of the inputs needed for a potentially representative input set. Unfortunately, the bound, which clearly exists given the decision procedure for the first order theory of $\mathbf{Q}$ under $+$ (see Specker and Strassen [76]), would be exponential in t, the Time bound for the branching program; hence we do not readily obtain a representative set as for the pure branching models since it appears to take time t to decode each input.

But still the problem is of enough independent interest that it is worth pursuing. And, moreover, this gives me an opportunity to argue that even if a direct corollary does not follow, the proof method may generalize. The case in point is the Time·Space $= \Omega(n^2)$ lower bound established by Borodin, *et al.* [79] for sorting on comparison branching programs. This result is "too low-level" to employ the previously discussed simulation for inferring a meaningful lower bound in the general setting. Yet, in this case Borodin and Cook [80] were able to show that the proof method does generalize and a bound of $\Omega(n^2/\log n)$ was established to sort n integers, each of length $O(\log n)$. Hence in terms of input string length $m = O(n \log n)$, we have the time-space bound $\Omega(m^2/\log^3 m)$. The idea in producing this general bound was to take a fairly structured view of the input without giving up any generality. I should also mention that Yao extended the original $\Omega(n^2)$ result to linear branching programs (i.e. a more powerful structured model). Unfortunately, there are no comparable lower bounds for a set recognition problem, in either setting.

Before leaving this section, we should mention another important structured comparison based model, the Batcher comparator network (see Knuth [73]). The model consists of a network (or circuit) with one type of gate, a comparator, which takes $< x, y >$ on input and outputs $< \max(x, y), \min(x, y) >$. Pippenger and Valiant [76] study an extension of this model, called ordering networks, where comparators are replaced by gates

$$\text{computing i)} \quad f(x,y) = \begin{cases} 1 & x \geqslant y \\ 0 & x < y \end{cases}$$

$$\text{and ii)} \quad g(b,x,y) = \begin{cases} x & b = 1 \\ y & b = 0, \end{cases}$$

and then augmented by the usual Boolean operations. Both models can be studied with respect to Depth or Size (= number of gates) of the network. The merging problem is relatively well understood on both models, with $\log n$ Depth and $n \log n$ Size being asymptotically necessary and sufficient. The sorting problem is relatively open. In particular, we know, $\Omega(n \log n) = \text{Size} = O(n \log^2 n)$ and simultaneously $O(\log^2 n)$ Depth on both models. For Depth alone the model is more critical. The conjecture is that sorting requires $\Omega(\log^2 n)$ Depth on comparator networks, whereas the results of Muller and Preparata [75] show that $O(\log n)$ Depth is sufficient (with Size $= O(n^2)$) for ordering networks. However, this raises the question as to whether or not we can simultaneously achieve $O(\log n)$ Depth and quasilinear Size (i.e. $O(n \log^k n)$). (For a much more powerful non oblivious parallel model, namely a comparison tree with n comparisons permitted in parallel, Valiant [75] can derive such simultaneous bounds—see also Preparata [78]). Even for comparator networks, there is no proof that Size $\cdot$ Depth $= \omega(n \log^2 n)$. This same issue concerning sorting on ordering networks can be viewed in the general setting of Boolean circuits which are to sort n numbers, each of binary length $O(\log n)$.

The Size-Depth problem for sorting (in contrast to Time-Space) seems to have a very interesting complexity behaviour, also observed for a variety of other problems involving simultaneous resource bounds. This behaviour is as follows: An optimal bound for measure 1 (say Depth $= O(\log n)$) can be achieved when the bound for measure 2 is essentially pessimal (say Size $= O(n^2)$), whereas by relaxing the measure 1 somewhat (to $O(\log^2 n)$) we can get good (i.e. quasilinear) measure 2 bounds. At the other extreme, an optimal measure 2 bound, say if $O(n \log n)$ Size were possible, seems to be achievable only with an essentially pessimal measure 1 bound. As other examples consider Space (as measure 1) and Time (measure 2) for the problems of the median (see Munro and Paterson [78] for the upper bound) and the string pattern matching problem (see Galil and Seiferas [77] for the upper bound). This latter problem exhibits a more quantitative statement of the behaviour, namely that with approximately k registers (which in our terms would be $k \log n$ Space since each register holds a pointer) one can solve the string pattern matching problem in Time

$O(n^{1+1/k})$. (Pippenger [personal communication] has recently shown that a similar upper bound can also be achieved for the sorting problem in the context of ordering networks.) This same quantative behaviour has a corresponding lower bound for the (structured) simulation of linear recursion schemes by flow-chart schemes that was referred to in Section I.

III. Arithmetic Models — Algebraic Complexity

I would now like to turn attention to the complexity of arithmetic problems, and to the straight line or circuit model with operations $+$, $-$, $\times$ (and perhaps $\div$). Fortunately, I need not pursue this topic in too much detail since Valiant [80] in this conference will be addressing just this topic. Indeed, Valiant [79*a*] and [79*b*] has always provided compelling evidence for the importance of the interrelation between a structured problem setting (algebraic complexity) and the general theory. The correspondence between algebraically structured arithmetic circuits computing (say) formal polynomials in $F[x_1, ..., x_n]$ and general Boolean circuits computing Boolean functions of n variables is readily apparent. In the former, gates represent the ring operations $(\times, +, -)$ and the inputs are the (indeterminates) $\{x_i\} \cup F$, while in the latter, the gates represent some basis set of Boolean operations (say $\wedge$, $\vee$, $\neg$) and the inputs are the (Boolean variables) $\{x_i\} \cup \{0 \text{ or false}, 1 \text{ or true}\}$. Since $\vee$, $\wedge$, $\neg$ can be easily simulated by $+$, $-$, $\times$ when restricted to $\{0, 1\{$ (e.g. $x \vee y$ by $x + y - x \times y$), positive results for the arithmetic case often carry over immediately to the Boolean setting. The usual measures of complexity are SIZE (= number of gates = sequential Time complexity), DEPTH (= length of longest path in the circuit = parallel time complexity) and FORMULA SIZE (= number of gates in a circuit having fan-out one; i.e. a formula). One of the first (pair of) results that demonstrated to me the importance of keeping this correspondence in mind, is the relating of the FORMULA SIZE and DEPTH measures. Independently, Spira [71] (for the Boolean case over any basis) and Brent [74] showed how to convert any formula of size m to an equivalent formula (and hence circuit) of depth $O(\log m)$; the converse that any circuit of depth d can be converted to a formula of size 2^d is immediate. It is interesting to note that the Spira result seems to depend intrinsically on the Boolean domain, whereas the Brent result is proven in a more abstract setting using only that $\times$ (resp. $\wedge$) distributes over $+$ (resp. $\vee$). Then, here again, is a situation where a result need not