Astérisque

L. G. VALIANT

Space-time tradeoffs in computations

Astérisque, tome 38-39 (1976), p. 253-264

<http://www.numdam.org/item?id=AST_1976__38-39__253_0>

Société Mathématique de France
Astérisque 38-39 (1976) p.253-264

Space-Time Tradeoffs in Computations

L. G. Valiant,
Centre for Computer Studies,
University of Leeds,
Leeds, U.K.

In everyday computing experience one can often speed up programs at the expense of using extra storage space. Alternatively, one can often reduce the storage space used at the expense of increasing computing time. It is natural to wonder whether such phenomena are isolated or whether they correspond to some general computational laws.

Our aim here is to show how such questions are formulated, to survey the few results that are now available, and to illustrate the fundamental and wide-ranging significance to computer science that further progress on these problems, in whichever direction, would have.

1. Computational Models

In order to ask precise quantitative questions about computations we have to commit ourselves to a specific machine model. In each context it is desirable to use a model that captures the intuitive notions that we are trying to describe as closely as possible. However, it is also

essential that the model be simple enough to be suitable for analysis.

For discussing the time and tape requirements of *discrete* computations the most widely-used model is the *multi-tape Turing machine* (TM). This consists of a read-only input tape, a finite number of read-write work tapes, and a finite-state control that is the program (as defined for example, in [11]). We say that the *time-complexity* of a TM is f(n) if for all n symbol inputs the computation halts after f(n) steps of the machine. The *space- complexity* is g(n) if for all n symbol inputs at most g(n) work-tape squares are ever visited.

The attraction of the TM model is that, while it is very simple to define and deal with, these two complexity measures for it are not gross overestimates in comparison with other reasonable models. In fact, in reprogramming an algorithm that takes f(n) steps on another model, onto a TM, one never has to increase the time complexity more than polynomially (i.e. not to more than $(f(n))^k$ for some k, ($k \leq 2$ usually)). Space complexity is even more robust, being usually essentially invariant under changes of machine model.

The above considerations show that by resolving complexity questions for TM computations we may expect to obtain results that have *universal* machine-independent significance. The evidence for this universality is analogous to Church's Thesis, resting, on the one hand, on a concensus about which are the relevant reasonable models, and, on the other, on proofs that these are equivalent to within *efficient* translations [1].

2. Complexity Classes

Define TIME(f(n)) to be the class of computational *problems* for which there is a TM of time-complexity f(n), and SPACE(f(n)) to be the class of problems for which there is a TM of space-complexity f(n). Fundamental facts about the time and space requirements of computations can be expressed by inclusion relations among such complexity classes. For example, the following relations, for arbitrary f(n), summarise the most obvious aspects of the time-space problem:

$$(X_0): \qquad \text{TIME}(f(n)) \subseteq \text{SPACE}(f(n)).$$

$$(Y_0): \qquad \text{SPACE}(f(n)) \subseteq \bigcup_{c=1}^{\infty} \text{TIME}(c^{f(n)}).$$

The first expresses the fact that in f(n) steps at most f(n) tape squares can be visited, the second that if the tape alphabet is of size c then after about c^x steps any computation on x tape squares must start repeating configurations.

The only progress on whether, and by how much, these relations can be tightened, has been achieved recently. It is shown in [10] that

$$(X_1): \qquad \text{TIME}(f(n)\log(f(n)) \subseteq \text{SPACE}(f(n)),$$

and that for any suitable increasing $\delta(n)$ (e.g. logloglog n)

$$(Y_1): \qquad \text{SPACE}(f(n)) \not\subseteq \text{TIME}(f(n)\log(f(n))/\delta(n)).$$

We shall return to these results in Section 4. Here we merely observe that the logarithmic factor is too fine to make these results necessarily universal in the sense of the previous section.

However, we can pose analogous questions in a manner that is coarse enough to ensure universal machine-independence significance. Let $\text{PTIME} = \bigcup_{k=1}^{\infty} \text{TIME}(n^k)$, $\text{PSPACE} = \bigcup_{k=1}^{\infty} \text{SPACE}(n^k)$, and $\text{PLOGSPACE} = \bigcup_{k=1}^{\infty} \text{SPACE}([\log n]^k)$. We can now ask the following questions:

(X): PTIME $\subseteq$? PLOGSPACE, and

(Y): PSPACE $\subseteq$? PTIME.

It is widely accepted that (X) and (Y) are two of the most fundamental open problems of computer science. As indicated below, this belief is based both on the relevance of these problems to resolving the inherent computational complexity of numerous particular computational tasks, and also to their relationship to other general questions.

It is appropriate to start by mentioning some basic types of results that shed light on several questions raised by the above formulation itself. Historically the first are perhaps the *hierarchy* theorems obtained by diagonalization arguments (e.g. [19]). A fundamental result is that for suitably well behaved f(n) and g(n), if

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

then $\text{SPACE}(f(n)) \not\subseteq \text{SPACE}(g(n))$. A similar proper inclusion result for time classes is provable under the stronger restriction that $\lim(f(n)\log(f(n))/g(n)) = 0$. Such facts contrast with *linear speedup* theorems which state, for example, that $\text{TIME}(f(n)) = \text{TIME}(kf(n))$ for any positive constant k.

Another type of result, obtained by *padding* or *translational* arguments. gives logical implications among relations between complexity classes. A typical result that holds for all suitably well behaved f,g and h, where $h(n) \geq n$, is that

$$\text{TIME}(f(n)) \subseteq \text{SPACE}(g(n)) \Rightarrow \text{TIME}(f(h(n)) \subseteq \text{SPACE}(f(h(n)). \quad (1)$$

The value of diagonalization and padding arguments is that they can be used to reduce the potentially unlimited number of possible questions about complexity classes to a small number of distinct types (see [2] for such applications). The following is an easy application: Suppose (X) is true. Then by (1) it is immediate that $\text{TIME}(2^n) \subseteq \text{SPACE}(n^k)$ for some k. But from the hierarchy theorem for time we have that $\text{TIME}(2^{n/2}) \subsetneq \text{TIME}(2^n)$. The two relations together imply that $\text{SPACE}(n^k) \not\subseteq \text{TIME}(2^{n/2})$. We have therefore deduced that if (X) is true then (Y) must be false. However, no converse of this is known and since it is widely conjectured that both (X) and (Y) are false, the two problems can be treated as distinct.

In conclusion we mention the following well-known and immediate connection with the Cook-Karp problem [4][13]: "if Y is true then P=NP". This suggests that in attempting to disprove P=NP, one ought to try to disprove Y first.

3. Complete Problems

A tantalising aspect of these general questions, that has been discovered only recently, is that whichever way they are resolved, they will settle important complexity questions for many specific natural computational problems. Conversely,

this means that it is sufficient to resolve these questions for these particular problems in order to determine the truth of the general assertions (X) and (Y).

The specific problems in this category are called *complete* in analogy with related phenomena in the theory of recursive functions. We say that a problem A in PSPACE is complete for (X) if it is provable that "A ϵ PTIME if and only if PTIME = PSPACE". Intuitively, this means that A is of about maximal complexity among all the problems computable in polynomial space. It turns out that many *natural* problems are complete for (Y). These include

(i) Equivalence of regular expressions [20].

(ii) Validity in the first order theory of equality [20].

(iii) Shannon switching game on vertices [7].

(iv) Context-sensitive recognition.

The last follows from the classical result of language theory that any TM that uses linear space can be encoded into a context sensitive language [14]. Proofs that the other problems are complete are of an analogous nature. However, the discovery that TM computations can be encoded into such diverse problems was a highly significant and surprising step with far reaching consequences [15].

Natural complete problems for (X) are also known. These are problems in PTIME with the property that they belong to PLOGSPACE if and only if PTIME = PLOGSPACE. These include "path systems" as defined in [5], and the emptiness of context-free languages [12].

Besides such specific problems, relationships with fundamental questions about other computational models are

known. A particular example is the size-depth problem for combinational Boolean circuits. It is observed in [3] that if (X) is true then it follows that for any circuit with n gates, there is an equivalent circuit of depth $[\log(n)]^k$ (for some fixed k). The best bound on depth that has been proved for this is $O(n/\log n)$ [17].

4. Results on Space-Time Tradeoffs

We shall now outline the results (X_1) and (Y_1) stated in Section 2. We start by considering a different computational model: A *straight-line program* is a sequence of instructions each of the form x:= F(y,z), where x, y and z are distinct elements from a set of variables, with the restriction that x cannot occur in any preceding instruction in the sequence. The domain of the variables and the set of operations from which each F can be chosen, are immaterial.

With each straight-line program a directed acyclic graph can be associated by identifying each variable u with a node $\bar{u}$, and each instruction x:= F(y,z) with a directed arc from $\bar{y}$ to $\bar{x}$ and one from $\bar{z}$ to $\bar{x}$. Directed paths in the graph therefore represent the flow of information in the program.

The execution of such a program can be regarded as a game on the graph with pebbles. (This game was introduced in [16] in the context of program schemas). The aim is to place a pebble on a distinguished output node using only the following types of moves:

(i) a pebble can be placed on a node either if it has no ancestors, or else if there are pebbles already on both of its immediate ancestors.

(ii) a pebble can be removed from any node (at any time).

Each pebble corresponds to a storage location. Placing a pebble on a node corresponds to storing the current value in the location. Rule (i) ensures that unless the variable is an input variable, the values of the variables from which it is computed are currently available in storage. Removing a pebble corresponds to freeing a storage location (with the possibility of recomputing it later).

Clearly the execution of the program defines a sequence of pebble moves all of type (i). However, any sequence of pebble moves that reaches the output node corresponds to a different but *equivalent* straight-line program. The question therefore arises as to whether some of these equivalent programs require substantially fewer pebbles. This is answered by the following result [10].

Theorem: Any node of a directed acyclic graph with n nodes and indegree k can be reached using $c_k n/\log_2 n$ pebbles.

This can be regarded as a space-time tradeoff, showing as it does that space can be reduced by a logarithmic factor at the expense of increasing the length of the straight-line program (possibly exponentially). The longer program saves storage by judiciously forgetting information and recomputing it later.

The proof given in [10] of this result implies (after a slight modification) an asymptotic upper bound on c_k of 4 for all k such that $\log k = o(\log n)$. Paterson has observed that if the constructive splitting strategy of [17] is used in the proof, then this can be improved to 2. (N.B. in the above application the case k = 2 is sufficient).

Paul, Tarjan and Celoni [18] have shown that the above

theorem is optimal. The graphs that are shown to require $O(n/\log n)$ pebbles are defined in terms of subgraphs for which only existential proofs are known. However, the universal graphs introduced in [21] are natural constructive examples that are known to require at least $O(n/(\log n)^2)$ pebbles.

Returning now to Turing machines, we note that each computation history can be regarded as a straight-line program. The difficulty, however, is that this graph may depend on the input and therefore cannot be determined beforehand. As is shown in [10], this can be overcome and it is possible to simulate any time $f(n)$ TM by a space $f(n)/\log(f(n))$ TM. The trick is to split up the computation into "large" blocks so that the graph of a computation is sufficiently "small". This ensures that the overheads of guessing the correct graph and the correct pebbling strategy add up to no more than a lower order term as compared with the space required to store the "contents of the pebbles".

The relation (X_1) follows from the above. The relation (Y_1), which can be re-interpreted as a nonlinear bound on the complexity of context-sensitive recognition, can be deduced by a diagonalization argument.

We conclude by mentioning that analogous results do exist for more restricted models. For one such model it has been shown that (X) is false [6]. Another example is the one-tape TM for which it is known that $\text{TIME}(f(n)) \subseteq \text{SPACE}(\sqrt{f(n)})$ [9]. However, one-tape machines are very restrictive in the sense that it can be easily shown that some problems (e.g. palindrome

recognition) that require $O(n^2)$ time on them can be computed much more efficiently (i.e. $O(n)$) on other models [11]. Since this has not been demonstrated for multi-tape TM's, even such apparently weak results as (X_1) and (Y_1) are of new significance.

References

[1] Aho, A.V., J.E. Hopcroft, and J.D. Ullman. The Design and Analysis of Computer Algorithms, Addison-Wesley, 1974.

[2] Book, R.V. Comparing complexity classes. JCSS 9 (1974) 213-229.

[3] Borodin, A. Some remarks on time-space and size-depth Manuscript (1975).

[4] Cook, S.A. The Complexity of theorem-proving procedures. Proc. 3rd ACM Symp. on Theory of Computing (1971) 151-158.

[5] Cook, S.A. An observation on time-storage tradeoff. Proc. 5th ACM Symp. on Theory of Computing (1973) 29-33.

[6] Cook, S.A. and R. Sethi. Storage requirements for deterministic polynomial time recognizable languages. Proc. 6th ACM Symp. on Theory of Computing (1974) 33-39.

[7] Even, S. and R.E. Tarjan. A combinatorial problem that is complete in polynomial space. Proc. 7th ACM Symp. on Theory of Computing (1975) 66-71.

[8] Hennie, F.C. and R.E. Stearns. Two tape simulations of multi-tape machines. JACM 13 (1966) 533-546.

[9] Hopcroft, J.E. and J.D. Ullman. Relations between tape and time complexities. JACM 15 (1968) 414-427.

[10] Hopcroft, J.E., W.J. Paul and L.G. Valiant. On time versus space and related problems. Proc. 16th IEEE Symp. on Foundations of Computer Science (1975) 57-64.

[11] Hopcroft, J.E. and J.D. Ullman. Formal Languages and their Relation to Automata. Addison-Wesley, (1969).

[12] Jones, N.D. and W.T. Laaser. Complete problems for deterministic polynomial time. Proc. 6th ACM Symp. on Theory of Computing (1974) 40-46.

[13] Karp, R.M. Reducibility among combinatorial problems. In Miller, R.E. and J.W. Thatcher (eds.), Complexity of Computer Computations, Plenum Press (1972).

[14] Landweber, P.S. Three theorems on phrase structure grammars of type 1. Inf. and Control 6 (1963) 131-137.

[15] Meyer, A.R. and L. Stockmeyer. The equivalence problem for regular expressions with squaring requires exponential space. Proc. 13th IEEE Symp. on Switching and Automata Theory (1972) 125-129.

[16] Paterson, M.S. and C.E. Hewitt. Comparative schematology. Proc. of Project MAC Conf. on Concurrent Syst. and Parallel Compt. (1970) 119-127.

[17] Paterson, M.S. and L.G. Valiant. Circuit size is nonlinear in depth. TCS (to appear).

[18] Paul, W.J., R.E. Tarjan and J.R. Celoni. Space bounds for a game on graphs. Proc. 8th ACM Symp. on Theory of Computing (1976).

[19] Stearns, R.E., J. Hartmanis , and P.M. Lewis. Hierarchies of memory limited computations. Proc. of 6th IEEE Symp. of Switching and Automata Theory (1965) 191-202.

[20] Stockmeyer, L. and A.R. Meyer. Word problems requiring exponential time. Proc. 5th ACM Symp. on Theory of Computing (1973) 1-9.

[21] Valiant, L.G. Universal Circuits. Proc. 8th ACM Symp. on Theory of Computing (1976).