an exact solution for all problem instances, however, this is enough to permit exact solutions for a restricted class of instances if \( c \) is small enough. In general, such algorithms exhibit tractable time costs when \( c \) is a polynomial function of the problem size. When applied to UVS, they will achieve exact solutions when \( c = \Phi, \) and, as I have argued, for biologically plausible UVS, \( \Phi \) is (at most) a polynomial function of problem size (i.e., retinal size \( s \)). If \( \Phi \) isn't polynomially related to \( s \), UVS is NP-complete and these algorithms may not deliver exact solutions in tractable time.

2. To solve Tsotsos's problem in the M-feature class, it suffices (though it may not be necessary) to consider each nonempty subset of the set of features types independently, thus a factor of \( 2^M - 1 \) appears in this straightforward expression for a time-cost upper bound for the problem.

Author's Response

Is complexity theory appropriate for analyzing biological systems?

John K. Tsotsos
Department of Computer Science, University of Toronto, and Canadian Institute for Advanced Research, Toronto, Ontario M5S 1A4, Canada
Electronic mail: tsotsos@cis.utoronto.edu

Kube has raised very important issues regarding the relationship between formal complexity theory and physically (and thus biologically) realizable systems. His comments have prompted a deeper and important discussion of current algorithms and of the nature of biological plausibility for solutions to problems of visual information processing.

Let me deal with several small points first before commenting on the key point of Kube's criticism. In my reply to Kube's first commentary, the following changes and comments should be made:

1. The line, "Moreover, by definition the relationship cannot be polynomial. Kube's comment does not fit the definitions," on page 164 is incorrect.

2. There is a typographical error at the top of page 461. The sum "\( \Theta + |l| \)" should have been the product, \( \Theta \cdot |l| \).

3. The statement that Knapsack's only solution is looking through all subsets was borrowed from a general source, as I point out on page 460, and I thought that this was a good way of making the point to a noncomputational audience. This in effect sweeps much under the rug - but I thought that it would have no consequence for the outcome of the work. As Kube points out, there is no net effect on the final result; it is important, however, that these issues be brought to the surface.

4. I did not show that the unbounded problem is a natural one. There is a class of figures where you indeed "see" to search all subsets of the image and where you do not know the target in advance, and the \( diff \) and \( corr \) functions could correspond to the computations that lead to the "ahh" that is experienced when you figure out what is in the scene. The reader may have seen some of these figures: the Dalmatian sniffing at leaves, the incomplete figures of Leeper (1935), the horseman of Verville and Cameron (1946) - even children's hidden picture exercises - all these seem to involve joining arbitrary collections of blobs, colors, and line segments until you sees some identifiable structure. It is not provable that all of these require inspection of all subsets, but it is clear that they require the inspection of a large number of arbitrary subsets in the general case.

5. I was mistaken to think that dynamic programming in general is not easily parallelizable. It is indeed parallelizable (mostly in its approximation version); there seem to be no algorithms for the specific version of Knapsack that I use, however. Several of these algorithms will be mentioned later in this paper. [See also Houston & McNamara: "A Framework for the Functional Analysis of Behavior" BBS 11(1) 1988, and Clark: "Modeling Behavioral Adaptations" BBS 14(1) 1991.]

I will now directly address Kube's concerns in his contribution to Continuing Commentary. He focuses on three main issues: What is biological plausibility for a theory of perception and how does this relate to my theory? What is the relationship between my theory and the available parallel solutions to the Knapsack problem? What is the dependence of my theory on the NP-Completeness of UVS (unbounded visual search)? Each is addressed in turn.

Biological plausibility does not imply polynomially large image values. Here is the crux of the debate: Is formal complexity theory appropriate for analyzing physical systems? On the one hand, there is a purely theoretical discussion of the complexity of a class of problems, visual search. Physical limitations, however, real do not enter into the discussion, just as they do not enter the discussion in any theoretical complexity argument (e.g., see Gopalkrishnan et al. 1991). The discussion there does not deal with physical realizability and the authors admit exponentially large values explicitly for the Knapsack problem without concern for how exponentially large numbers can be represented physically. This is true in any discussion of complexity. It is also true that algorithms with polynomial complexity are believed to be "good" and those with exponential complexity "bad"; yet there is an infinite number of values of exponents and variables that would lead to the exact reverse when an algorithm is physically realized. Consider simply the following pair of functions: \( O(n^\alpha) \) and \( O(2^{\alpha n}) \). It is easy to see that there is an infinite space in which the polynomial function's actual value is larger than the exponential depending on the values of the constants \( n \) and \( \alpha \). And of course there is an infinite number of such function pairs that we may compare. Early complexity theorists understood this problem, yet they claimed that polynomial functions with "bad" behavior do not occur in practice, nor do exponential functions with "good" behavior. Thus, the search for polynomial and subpolynomial complexity functions is the driving goal of theory.

An important issue seems to have been forgotten, however: If the practice of complexity analysis is to lead to tangible benefits then the theorems must lead to algorithms that are physically realizable and the physical realization must be in some way be "better" than others with respect to time or space efficiency. No matter what the time and space complexity functions, there is an infinite space of possible variable values or problem sizes that will not be practically realizable. The fact that all computers have finite memories is sufficient to guarantee this. One cannot in practice take infinite time to read or load in infinite Turing machine tape. Engineering design
specifications always impose constraints. The amount of memory may be limited by power consumption or cost; the number of processors is likewise constrained; real-time response places a hard constraint on time complexity and thus on problem size. These constraints cannot be ignored in any complexity discussion that may eventually be used to solve real problems. And the whole point of complexity theory is formally to provide insights on the relative difficulty of real problems. Yet virtually all theoretical discussions do exactly this.

Let me move on to a specific physical system, the brain. Biological plausibility means that the functions for time and space complexity of any algorithm that we claim performs some information processing task in the brain permit only values of their variables that lead to brain-sized space requirements and behaviourally confirmed time requirements. Issues of polynomial versus exponential time do not enter the discussion of biological plausibility at all.

What do I mean by biological plausibility? Throughout my target article I attempted to describe these criteria (and where they appear in the target article is noted below), but perhaps a summary and restatement is appropriate. A number of constraints must be satisfied:

1. Computation is performed in parallel, and responses across a layer of parallel processors all seem to appear simultaneously, with no apparent delays.

2. Solutions should require significantly fewer than about 10⁹ processors operating in parallel (p. 432).

3. Processor average fan-in and fan-out should be about 1000 overall (p. 437).

4. Solutions should not involve more than 100 or so sequential processing steps (p. 424).

Although we know about such constraints, in the beginning, "evolutionarily" speaking, it must have been impossible to make such a requirements specification; it can only be made in hindsight using the "experience" that Bellman (1954) talks about. I tried to capture the problem before making assumptions. Who would start off designing the very first vision system ever by first ensuring that the magnitude of the measurements was not an exponential function of the image set? I tried very hard not to make any assumptions about the data at that stage of the analysis.

It appears that knowledge of the magnitudes of the numbers in the unbounded visual search problem when accompanied by these constraints on the solution will not allow an acceptable dynamic programming solution, even if that knowledge alone may yield a parallel solution that is polynomial in time and in required processors. The strategy I use to demonstrate this point is to consider each of the current algorithms in turn in a "backwards" fashion, that is, given time and space complexity functions, what values of their variables will yield complexity estimates that satisfy the above constraints. Those values will show that only problems that are comparatively trivial and not representative of vision at any stage of processing in the brain satisfy the constraints. Note that since none of these algorithms proves a lower bound for the problem, it is still an open problem whether more efficient solutions exist.

The following should be noted regarding the comparisons in Table 1:

1. All logarithms are base 2.

2. The variables used represent:
   \( n \) the size of the input set, \( |I| = P \times M \)
   \( c \) max value of the constraint on the corr function, \( \delta/p^2 \)
   \( e \) allowable error for approximation algorithms, \( 0 < \epsilon \leq 1 \), that is, solution will be at least \((1 - \epsilon)\) times optimal.
   \( m \) the maximum value of the constraint on the diff function, \( \delta/p \)
   \( s \) the max value of the diff function
   \( M(d) \) the number of processors required to multiply two \( d \times d \) matrices in \( O(\log d) \) time. The determination on the minimum number is an open problem, it is known however that \( d^{2.5} \) suffice, and thus this will be used below.

3. for the approximation solutions, we set \( \epsilon = 0.1 \) thus a solution within 90% of the optimal is required.

4. I will use Kube’s estimates for the variables in the table below, so:
   \( m = 10^{10}/p \)
   \( c = 10^{20}/p^2 \)
   \( s = 10^{10}/p \)

   For convenience and with no loss of generality, and in the interests of performing this comparison optimistically, I assume \( p = 1 \).

These are very large numbers. I have also done some of the computations in the table below, solving for the values of the variables \( m \) and \( c \) to see exactly what range could be accommodated by those algorithms with biological plausibility. It is clear that these algorithms are seriously out of range for plausibility. We seek the largest values of the variable \( n \) so that the biological constraints will not be violated. The complexity function is set to the maximum values of biologically plausible time and processors (100 and 10⁶, respectively) and then we solve for \( n \). For example, if the time complexity is \( \log n \), and the time constraint is 100, then the value of \( n \) that yields a time complexity of 100 would be \( 2^{100} \) or \( 10^{100} \).

4. A number of different computing architectures are employed in the algorithms compared in the table. The following acronyms are used to denote those architectures:
   - **PRAM**: Parallel random access machine – each processor can access its local memory, access the shared memory and perform one standard operation.
   - **CREW-PRAM**: Concurrent read exclusive write on a parallel random access machine.
   - **EREW-PRAM**: Exclusive read exclusive write on a parallel random access machine.
   - **CRCW-PRAM**: Concurrent read concurrent write on a parallel random access machine.

It is not clear what the correspondence is between the types of computer architectures and neural processing. The one used in my target article corresponds to a **CREW-PRAM** model. In addition, none of these algorithms is specific for the version of Knapsack on which visual search is based. Thus the comparison is not exact, although we may assume that it is approximate.

5. If any of the algorithms yields a biologically plausible solution, then the minimum value of \( n \) across the two right-hand columns of the table will be at least 1000 using the same criterion for \( n \) as in the target article.

6. Finally, the references cited in the table correspond to: (1) Dantzig (1957), (2) Bytter (1988), (3) Teng (1990), (4) Peters and Rudolph (1987), and (5) Gopalkrishnan et al. (1989).
Table 1. Comparing the biological plausibility of a number of parallel algorithms for the Knapsack problem showing that none of the currently known parallel solutions to Knapsack even approaches satisfying the requirements laid out for biological plausibility.

<table>
<thead>
<tr>
<th>Problem and Reference</th>
<th>Time Complexity</th>
<th>Processor Complexity</th>
<th>If time = 100, Approx. n =</th>
<th>If processors = 10⁶/10⁹, Approx. n =</th>
</tr>
</thead>
<tbody>
<tr>
<td>0. Sequential dynamic programming (1)</td>
<td>( nm )</td>
<td>1</td>
<td>&lt;1 (( n = 1000 ) for ( m = 0.01 ))</td>
<td>N/A</td>
</tr>
<tr>
<td>1. Parallel dynamic programming on a CREW-PRAM (2)</td>
<td>( \log^2 n )</td>
<td>( n^6 / \log n )</td>
<td>1024</td>
<td>10 / 42</td>
</tr>
<tr>
<td>2. Parallel dynamic programming on a CREW-PRAM (2)</td>
<td>( \log n )</td>
<td>( n^4 )</td>
<td>10⁶⁰</td>
<td>31 / 178</td>
</tr>
<tr>
<td>3. Parallel approximate 0–1 Knapsack on a CREW-PRAM (3)</td>
<td>( \log n \log(n/\varepsilon) )</td>
<td>( n^2 / \varepsilon )</td>
<td>355</td>
<td>316 / 10,000</td>
</tr>
<tr>
<td>4. Parallel integral knapsack #1 on a PRAM (3)</td>
<td>( \log mn + \log^2 m )</td>
<td>( mn / \log mn + M(m) )</td>
<td>&lt;0 (( n = 1000 ) for ( m = 512 ))</td>
<td>&lt;1 / &lt;1 (( n = 1000 ) for ( m = 250 / 3950 ))</td>
</tr>
<tr>
<td>5. Parallel integral knapsack #2 on a PRAM (3)</td>
<td>( \log mn + \log (m/s) \log^2 s )</td>
<td>( mn / \log mn + M(s/m) / \log m/s )</td>
<td>&lt;0</td>
<td>&lt;1 / &lt;1</td>
</tr>
<tr>
<td>6. Parallel integral knapsack #3 on a PRAM (3)</td>
<td>( \log mn + \log m \log n )</td>
<td>( mn / \log mn + m^2 n / \log m \log n )</td>
<td>&lt;1 (( n = 1000 ) for ( m = 296 ))</td>
<td>&lt;1 / &lt;1 (( n = 1000 ) for ( m = 275 / 10,000 ))</td>
</tr>
<tr>
<td>7. Parallel approximate 0–1 Knapsack on a CREW-PRAM (4)</td>
<td>( \log^2 n + \log(n/\varepsilon) )</td>
<td>( n^3 / \varepsilon^2 )</td>
<td>355</td>
<td>21 / 215</td>
</tr>
<tr>
<td>8. Parallel optimal 0–1 Knapsack on a CREW-PRAM (5)</td>
<td>( \log n \log c )</td>
<td>( nc^2 )</td>
<td>2 (( n = 1000 ) for ( c = 1048 ))</td>
<td>&lt;1 / &lt;1 (( n = 1000 ) for ( c = 31 / 1000 ))</td>
</tr>
<tr>
<td>9. Parallel approximate 0–1 Knapsack on a CREW-PRAM (5)</td>
<td>( \log^4 n + \log^3 n \log(1/\varepsilon) )</td>
<td>( n^{3.5} / \varepsilon^{1.5} )</td>
<td>3</td>
<td>63 / 1000</td>
</tr>
</tbody>
</table>

Now let me turn to the strategy I chose in my target article. I defined a problem I called unbounded visual search and proved that it was NP-Complete. I then claimed the fact that this problem is exponential leads to a contradiction because I related it directly to the experimental paradigm of visual search. The unbounded problem, that is, when the target is unknown, is the kind of task humans perform all the time simply in interpreting their everyday environment. The bounded version, with a known target, is known to require time linear in the size of the test display. The contradiction arises for the unbounded problem: Whereas humans have no difficulty in managing their existence in unbounded situations, they cannot be solving a problem that requires exponential time. The problem must be re-shaped and resources must be optimized in order to overcome this contradiction. Then I proceeded to show that a particular set of approximations and optimizations leads to the right kind of space and time complexity functions so that the resulting algorithm is biologically plausible. An important trick is to define the problems, unbounded and bounded, in such a way as to permit the simple addition of task knowledge to lead to a linear solution. Up to here, I do not believe Kube would have any objections save one: Because we require humans to solve the problem, we know the size of the problem and the magnitude of the numbers used, and thus, the dynamic programming approach can be used to produce a polynomial time solution.

In my target article, I use Stockmeyer and Chandra's (1979) argument why some problems defy computation. They claim that some problems are so large that even if the entire universe were the processor, they would not be computed within the age of the universe. This is precisely the same argument I am making here but on a different scale, a human scale. Physical realizability plays a crucial role in complexity theory. If complexity theory could be refuted by a change in scale, then the theory has no value.

Although Kirousis and Papadimitriou have indeed
shown that "a" visual problem is NP-Complete (1985), they did not show this for a problem that is a numeric one, which most vision problems that deal with real images are, and for a problem that lends itself directly to comparison with experimental data. As to the comment about the exponentially many templates, this leads to a problem that violates the constraints for biological plausibility laid out earlier. If there is a fixed memory, then the search through the templates requires time linear in the memory size.

But is there a "dependence" on the NP-Completeness of the problem for the rest of the work? Here are some of the reasons this proof has important consequences:

a. NP-Completeness shows that the problem requires in the worst case execution time exponential in the problem size.

b. Because for my problem size is defined as a function of the product of P and M, it leads to the exponential in M at the end of the analysis. Although I have no proof that humans treat M in an exponential fashion, it is clear that the feature dimension is handled differently from the number of items in visual search experiments. A small piece of evidence is presented in my first-round Author's Response.

c. NP-Completeness theory provides guidelines on how to deal with NP-Complete problems, namely, attacking the problem by reducing the large natural variable.

d. NP-Completeness shows us that approximations and optimizations are necessary; pure massive parallelism is not sufficient. Dynamic programming (DP) is one approach.

e. One major problem with the DP approach is that it does not explicitly permit the variable P to be handled differently than the variable M.

NP-Completeness provides an important formal basis for this discussion. The key aspect of the theorem is not the proof (which is simple from a complexity perspective). It is in the formalization of the problem so that it becomes amenable to proof (and, by the way, to the DP solution as well) in the first place. Vision does not have this kind of theoretical foundation and that is where I feel the contribution of this portion of the work lies.

One final point: We cannot assume that evolution finds optimal solutions in the same sense that complexity theory seeks. Evolution finds satisficing solutions and it is these solutions I am attempting to find. It would be an uninteresting conclusion if complexity theory applied only to artificial computational problems and not natural ones.

NOTE

1. The algorithms cited all deal with integers. To make the comparison correct, we must convert the real numbers in our version of visual search into integers. This is easily done by dividing each instance of a number by the precision $\rho$. Thus, $\theta$, which the product of two numbers via the $\text{corr}$ function, must be divided by $\rho^2$.

ACKNOWLEDGMENTS

I thank Joe Peters, Patrick Dymond, and Bart Selman for important discussions. The author is the CP Fellow of the Canadian Institute for Advanced Research.

References


