The Relevance of Parameterized Computational Complexity Theory to Cognitive Modeling

H. Todd Wareham

Department of Computer Science
University of Victoria
Victoria, BC Canada
harold@csr.uvic.ca

August 2, 1996

Abstract

- In order to evaluate the utility of a measure M of the power of cognitive models, it is useful to answer the following questions:

  (1) What is M?

  (2) Why is M a good measure of the power of a model?

  (3) What is the power of a model under M?

  (4) How does one measure the power of a model under M?

  (5) What are the sources of power in a model under M?

  (6) How does one isolate the sources of this power?

- Focus on computational complexity as a measure of model power.
- Within this framework, the recently developed theory of parameterized computational complexity [4] can be used to isolate the sources of model power.
- This discussion is illustrated by an analysis of the sources of power in Declarative Phonology [7], a constraint-based model in linguistics.

What is computational complexity?

- A *(computational) problem* P is a relation between inputs and outputs; an *algorithm* A for P is a set of instructions relative to some computer which, for any input of P, computes the appropriate output. We say that A *solves* P.

- The *computational complexity of an algorithm* is a function bounding the amount of some computational resource, e.g., processor time or memory, required by the algorithm to solve its associated problem as a function of input size.

- The *computational complexity of a problem* is the set of computational complexity functions of its associated algorithms; this set is often represented by the "best" function in the set.

- A problem is *tractable* if it has a "good" algorithm, e.g., resource requirements are bounded above by a polynomial in the size of the input; else, it is *intractable*.

- Computational complexity theory [5] consists of various techniques for assessing the computational complexity of both algorithms and problems.

Why is computational complexity a good measure of the power of a cognitive model?

(1) Computational complexity has useful properties:

   (a) Computational complexity is asymptotic, i.e., resource-bounds hold over all input sizes; hence, a measure of power automatically "scales up" to systems of realistic size.

   (b) Computational complexity is machine-independent; hence, a measure of power can be rephrased relative to an algorithm running on any underlying neural architecture.

(2) Every aspect of a model contributes to that model's computational complexity; hence, computational complexity provides a common framework for examining the sources of a model's power in terms of any aspect of that model.

(3) Cognitive systems are computing systems; hence, computational complexity provides a common framework for examining such systems and their models, as well as techniques for refining these models, e.g., "cognition complexity games" [6, 8].

What is the computational complexity of a cognitive model?

- The computational complexity of a cognitive model is the set of complexities of the problems associated with that model, each of which is in turn is a set of complexity functions of algorithms.
- Typically, will focus on one problem associated with a model; results for this problem give a lower bound on the complexity of the model.
- Measuring model power as a set instead of a single value is a reflection of our ignorance of how the cognitive system being modeled operates in practice; as details about the actual system are discovered, the focus narrows to problems and algorithms that are consistent with these details.

- In the example below, a model of phonological processing is examined in terms of the speech-production problem, e.g., deriving speech signal from lexical entries; there are other problems that are equally valid, e.g., deriving lexical entries from speech signal.

How can the computational complexity of a cognitive model be measured?

- Measure the computational complexity of a model by establishing the complexities of its associated problems.
- Can establish that a problem is tractable by exhibiting a "good" algorithm; how can we show intractability?
  - Define classes F and C of tractable and intractable problems such that it is either known or strongly conjectured that $F \subset C$.
  - If a problem P can be used to solve any problem in C then P is C*-hard*; if P is also in C then P is C*-complete.*
  - If P is C-hard then it is not tractable (modulo the strength of the assumption that $F \subset C$).
- Most famous theory of computational complexity is that for $\mathcal{NP}$-completeness [5], with tractable class $\mathcal{P}$ of problems that can be solved in polynomial time and intractable class $\mathcal{NP}$ of problems that can be checked in polynomial time.

What are the sources of computational complexity in a cognitive model?

- The sources of computational complexity in a cognitive model are the aspects of the model that are factors in the complexity functions of the algorithms associated with that models problems.
- Focus on sources of power that are responsible for non-tractable algorithm behavior.
- Useful to know all sources of complexity in a model because as details of the cognitive system are discovered, the focus narrows to those algorithms that are efficient relative to those details, e.g., given algorithms A and B with complexity functions $2^k l$ and l^k, A will be preferable when k is a constant much smaller than l.

- In the example below, possible sources of complexity are the number of constraints ($|A|$), the length of the derived surface structure (m), the size of the alphabet ($|\Sigma|$), and the maximum constraint context-size (c).

How can the sources of computational complexity in a cognitive model be isolated?

- Isolate the sources of computational complexity in a cognitive model by showing which aspects of that model are responsible for non-tractable behavior in algorithms for that model's problems.
- Cannot do this in classical theories of computational complexity; proofs of intractability do not say which factors are responsible.
- Can do this in the theory of parameterized computational complexity [4].
  - Defines two-part problem instances $\langle x, k \rangle$ in which k is called the *parameter*.
  - Has tractable class FPT of parameterized problems solvable in time $f(k) \cdot |x|^\alpha$, where f is an arbitrary function and α is a constant independent of k, and intractable classes $W[1], W[2], \ldots$ (the W*-hierarchy*).
  - If a k-parameterized problem P is $W[t]$-hard then k is not a source of non-polynomial time behavior in P (modulo the strength of the assumption that $FPT \subset W[t]$).

Constraint-Based Theories in Linguistics: Declarative Phonology

- Phonology concerned with the relation between deep (lexical) and surface (spoken) structures. Two ways of implementing this relation:

   1. **Rule-based**: Apply rewriting rules to deep structure to create surface structure.
   2. **Constraint-Based**: Apply constraints to set of candidate surface structures generated from deep structure to select surface structure.

- Focus on Declarative Phonology [7], which requires that selected surface structures satisfy all constraints. Assume following:

   (1) Set of candidate structures is a regular language generated by a finite state transducer T from a string u.

   (2) Constraints are specified as contextual DFA (CDFA). Each CDFA has an associated context-length c such that this CDFA applies to and evaluates each c-length substring of a candidate structure.

Declarative Phonology: Computational Problems

Solution Problem:

DECLARATIVE PHONOLOGY DERIVATION (DPD-SOL)
Input: A string u, a finite state transducer T, and a set of constraint CDFA A.
Output: Any string x such that y is generated by T operating on u and x does not violate A_i, $1 \leq i \leq |A|$.

Associated Decision Problem:

DECLARATIVE PHONOLOGY DERIVATION (DPD-DEC)
Input: A string u, a finite-state transducer T, a set of constraint CDFA A with maximum context-length c, and an integer m.
Question: Is there a string x of length m such that x is generated by T operating on u and x does not violate A_i, $1 \leq i \leq |A|$?

- Study decision problems because (1) decision problems easier to analyze and (2) results for a properly-formulated decision problem also apply to the associated solution problem.

Declarative Phonology: The Power of the Model

Theorem DPD-Dec is $\mathcal{NP}$-complete.

Proof (Sketch): Trivially in $\mathcal{NP}$. Show $\mathcal{NP}$-hardness by reduction from the following problem:

Longest common subsequence (LCS)
Input: A set of k strings $X_1, ..., X_k$ over an alphabet Σ, and a positive integer m.
Question: Is there a string $X \in \Sigma^*$ of length m that is a subsequence of X_i for $i = 1, ..., k$?

Transform instance of LCS into DPD-Dec as follows: Create constraint-set A by transforming each X_i into its corresponding subsequence-DFA by the construction of Baeza-Yates [1] and construct C to recognize all strings over Σ of length m. $\square$

Declarative Phonology: The Sources of the Power

	$\|\Sigma\|$		
Parameter(s)	Unbounded Value	Parameter	Fixed Constant
–	–	W[P]-hard	W[P]-hard
$\|A\|$	W[t]-hard for all $t \geq 1$	W[t]-hard for all $t \geq 1$	W[t]-hard for all $t \geq 1$
m	W[2]-hard	FPT	FPT
c	W[2]-hard	?	?
$\|A\|$,m	W[1]-hard	FPT	FPT
$\|A\|$,c	W[1]-hard	?	?
m,c	W[2]-hard	FPT	FPT
$\|A\|$,m,c	W[1]-hard	FPT	FPT

- Many of these proofs follow from the reduction in the previous theorem and the results in Bodlaender *et al.* [3].

Declarative Phonology: Implications

- The $\mathcal{NP}$-hardness of DPD-Dec shows that this problem is, in general, intractable; the parameterized analyses done so far show that this intractability can be isolated in the size of the candidate structure set.
- Consequences:
  1. Restricting candidate structure sets to regular languages does not guarantee tractability, cf. [2].
  2. Restricting maximum context-size or the number of constraints does not guarantee tractability; additional mechanisms must also be restricted to model phenomena that require unbounded-size constraints, e.g, vowel harmony.
- These results also apply to several other constraint-based phonological theories, e.g., KIMMO, Optimality Theory.
- Future research should focus on further restrictions of and different formalisms for expressing candidate structure sets and constraints.

References

[1] Richard A. Baeza-Yates. 1991. Searching Subsequences. *Theoretical Computer Science*, 78, 363–376.

[2] Stephen Bird and T. Mark Ellison. 1994. One level phonology: autosegmental representations and rules as finite automata. *Computational Linguistics*, 20, 55-90.

[3] Hans Bodlaender, Rod G. Downey, Michael R. Fellows, and H. Todd Wareham. 1995. The Parameterized Complexity of Sequence Alignment and Consensus. *Theoretical Computer Science*, 147(1-2), 31–54.

[4] Rod G. Downey and Michael R. Fellows. 1995. Fixed-parameter tractability and completeness I. Basic results. *SIAM Journal on Computing*, 24(4), 873–921.

[5] Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness.* W. H. Freeman and Company, San Francisco.

[6] Eric S. Ristad. 1993. *The Language Complexity Game.* MIT Press, Cambridge, MA.

[7] James M. Scobbie. 1991. Towards declarative phonology. In Stephen Bird (ed.) *Declarative Perspectives on Phonology*, pages 1–26. University of Edinburgh.

[8] John K. Tsotsos. 1993. The Role of Computational Complexity in Perceptual Theory. In Sergio C. Masin (ed.) *Foundations of Perceptual Theory*, pages 261–296. North-Holland, Amsterdam.

Papers of Interest

**Display Copies Only –
Please Do Not Take!**

Some of These Papers are Available on My Home Page:

http://www-csc.uvic.ca/home/harold/harold.html

Visit the Parameterized Complexity Home Page at:

http://www-csc.uvic.ca/home/harold/W_hier/W_hier.html