Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2007-026 July 22, 2007

Developmental Cost for Models of Intelligence

Jacob Beal

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

Developmental Cost for Models of Intelligence

Jacob Beal*
MIT CSAIL
jakebeal@mit.edu

We can evaluate models of natural intelligence, as well as their individual components, by using a model of hardware and development costs, ignoring almost all the details of biology. The basic argument is that neither the gross anatomy of the brain nor the behavior of individual cells nor the behavior of the whole poses sufficient constraint on the algorithms that might run within the brain, but that the process of engineering an intelligence under this cost model poses similar challenges to those faced by a human growing from a single cell to an adult. This will allow us to explore architectural ideas freely, yet retain confidence that when a system works, the principles allowing it to work are likely to be similar to those that allow human intelligence to work.

Neuroscience is not enough

Although our current knowledge of biology tells us a great deal about how human intelligence *cannot* be structured, we know little about the algorithms that actually make our intelligence work.

Let us start big, with our knowledge about the anatomy of the brain. By studying tissue structure, injuries, and intensity of blood flow, we know quite well the gross location of broad functions, like "vision" and "motor control." Other things, like "memory" or "language" are more slippery, with lots of different places involved in different ways. Knowing how much activity is going on in what places, however, tells us nothing about what computations are being executed or their relative importance.

To see why, consider a modern computer like the laptop I am using to write this. Modern computers spend almost all of their cycles either doing nothing or putting pictures on the screen. Editing this document occupies effectively zero processing power[1], and storing the text takes only a few millionths of the available storage.

If you scanned my computer's components for activity, correlating carefully with when I hit various keys, you could probably discover the hardware controlling the keyboard and hard drive. When you try to learn how the spell-checker works, however, you learn only that it depends on interactions between the processor, memory, the hard drive, the keyboard, and the graphics coprocessor. You may investigate further, determining that when interaction with the hard-drive is interrupted, no spell-checking occurs at all, while disrupting interaction with the graphics coprocessor allows spell-checking to run, but ineffectively. Still, the information about how it actually works is obscured by all the other things going on in the background.

We are in much the same position concerning the anatomy of the brain: we know quite a bit about what places are involved with various sorts of activity. What we do not have is a clear understanding of what computations are being executed in these places. The few exceptions are places like early vision, where the behavior is very closely coupled to input, or like the pituitary gland, where the behavior is highly specialized. Because of this, our knowledge of anatomy tells us virtually nothing about how the parts work together to produce human intelligence.

What about the other end of the scale, the behavior of individual cells? We know quite a bit about the various types of cells in our brains, and especially about the neurons where it appears that most of the computation takes place. We can even stick probes into brains and measure the behavior of some individual neurons. Knowing what an individual cell does, however, tells us nothing about the larger computation in which it is participating.

Again, consider trying to figure out how my spell-checker works. Sticking delicate probes into the guts of my poor laptop, you discover that there are lots of tiny parts. Although they all share common features, different parts have specialized to perform different functions: some parts switch on when their inputs are active, while others switch on when their inputs are not active. Some parts store values and repeat them again later, while others relay values across long distances. When you apply this knowledge to the spell-checker, however, you again come up blank. Each time you run the spell-checker, different parts of the memory are used, and although the same parts of the processor are used each time, the pattern of use is always different—the rest of the work the computer is doing keeps interfering in different ways. With some hard work, you can find the parts that test for equality, and show that when there are errors they behave differently from when there are none. Still, the information about how it actually works is obscured, both by the rest of the activity going on in the computer and by the fact that the

*This work is supported by NSF Grant #0534978
Copyright © 2007, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.
[1] I use `emacs`.

parts are so generic that they could be doing anything.

We are in much the same position concerning the behavior of neurons. We know quite a bit about what sort of computation an individual neuron can do, what a collection of interconnected neurons can compute, and how long a computation would take. The problem is that a large collection of neurons can potentially compute just about anything, and we know very little about what they actually are computing. Because of this, knowing about individual neurons tells us virtually nothing about how large numbers work together to produce human intelligence.

Finally, what about comparison to human behavior? By measuring reaction times, relative preferences, and other objective features of cognition, cognitive science has produced a great wealth of information about the range of behavior produced by actual human intelligences. Knowing what a complete human intelligence does, however, tells us nothing about what its components are or the types of functions they compute.

Once more, consider my spell-checker. Sitting down at the keyboard, you discover that hitting "Escape-$" causes it to tell whether a word is correct—a fine stimulus response! Comparing across computers and applications, you discover that the spelling reflex is universal (except for a few diseased individuals) and document a range of stimuli that activate the spelling reflex in different contexts, including "Command-:" and "F7." Some further analysis reveals universals and range of variations: for example, the first suggestion to correct "teh" is always "the," but the number and nature of other suggestions varies widely. Combining this with previous work on component activity, you can make some pretty good theories about how the keyboard and graphics coprocessor contribute to spell-checking, but the contribution of the processor and memory is still a mystery. For that, you start setting up much more precise experiments, timing how long spell-check takes to start and how long between words, varying the size of the document, the distribution of errors, and whether music is playing at the same time, and so on. You learn many facts, including that longer documents take longer to check, and that music can slow the process down, presumably by competing for a shared resource. But you end up stymied still, because both the memory and the processor are always interacting with one another, and every manipulation you apply still involves both equally.[2] In the end, all you know is that processor and memory work together to spell-check, but nothing about how they do so.

Measurements of human behavior have the same problem: we can measure the behavior of the whole system, and compare the results of different experiments in order to understand which parts are involved and how the behavior changes if a part is damaged. What we do not know is how the parts interact with one another to produce the observed behavior. Even apparently modular behavior might involve small but important interactions between many components. The exceptions are phenomena like optical illusions, where the behavior involves very specific resources and does not depend much on interaction. Moreover, we do not know how sensitive the observed behavior might be to small flaws in our models: an almost correct model missing one critical link may produce a much worse fit than a model which is completely incorrect, but well-tuned for a particular domain. Because of this, measurements of human behavior tell us virtually nothing about how the parts work together to create the behavior we observe.

[2]This will happen if the dictionary does not fit in the processor cache.

Development is Like Engineering

Thus far, I have basically argued that when we build a system that conforms to what we currently know about brain anatomy, individual neurons, and human behavior, there is not enough constraint for us to have confidence that it will be smart like a human. Moreover, if we build something that is not smart like a human, we do not know enough about how the parts should interact to figure out what we should change in order to make it smarter.

So again, here is the challenge we face as engineers: we need to build many different parts and connect them together to form something that is smart like a human. This is hard because the parts are complicated, so we need to work on them separately, and how to specify the goal is unclear, so we will make many mistakes in building them. Can we even dare hope that such a disjointed and flawed process can produce parts that learn to work together and be smart like a human? It seems to need a miracle.

Such a miracle has happened not once, but billions of times, every time a human grows from a single cell to an adult. First, because the various parts of the brain are far from each other (centimeters are huge in cellular terms!) the details of each part must develop largely independently. Our problems engineering an intelligence also lead to parts being developed largely independently, though for the engineer it is the complexity of the problem that causes separation in time and space (different engineers). Second, there is a large amount of variation during the development of the brain. In engineering, independent development of parts also leads to variation, because we do not understand the problem well enough to prevent our conception of the system from changing over time or between engineers.

Development overcomes these problems, regularly integrating the parts of the brain into a functional intelligence.[3] Yet we engineers have not yet been able to do the same as we struggle to build an intelligence.

I propose that there are organizational principles exploited by biology which we have not yet recognized and tamed to our purposes. What we want, then, is to place ourselves in a situation where we are wrestling with the problems that development overcomes, so that we are more likely to discover these missing pieces. As an additional benefit, if the systems we build are biologically plausible, then it is reasonable to guess that the organizational principles which allow

[3]Even most mentally ill or developmentally disabled people are basically functional examples of integrated human intelligence. It's just that our standards are very high and we notice even small differences in behavior.

us to integrate the pieces of our systems will be related to the principles that allow human development to work.

Hardware and Development Costs

What constraints shall we set for ourselves? On the one hand, we want simple constraints so that we can explore architectural ideas freely. On the other hand, our systems must not be obviously biologically implausible.

I propose to resolve this dilemma with a biologically-inspired model of hardware and development cost. This model will allow us to judge an individual part without knowing how it will be employed, by measuring the asymptotic cost as the capacity of the part increases.

This measurement, in turn, tells us what constraints there are on our ability to use the part as part of a broader explanation of intelligence. If a part is costly, we can only use a few small-capacity copies. If a part is cheap, then we may be able to use vast numbers or a few with vast capacities. Thus, for example, a part involved in handling words should have a low cost per word, because we need to handle thousands of different words, but a part managing our response to hunger could be costly because hunger is one of a very few direct survival urges.

I propose measuring two types of cost familiar to computer scientists: time complexity and space complexity. Because we are making an analogy to development, however, we must measure these costs both for a mature system and for its development. The relative importance of these costs differs from what we have become accustomed to on digital computers.

- **Time Complexity:** Biological systems are much slower than silicon systems, operating at frequencies ranging from kilohertz to millihertz depending on the mechanism, so time is a more precious resource. On the other hand, there is the opportunity for massive parallelism. The available time is generally long during development, ranging from weeks to years, but may be short during mature execution, on the order of seconds or less.
- **Space Complexity:** Space complexity refers to the amount of hardware comprising the mature system and the size of the program encoding its development. For the mature system, this generally means neurons, of which there are approximately 100 billion in the brain (approximately 20 billion in the neocortex), each connecting to an approximate range of 100 to 10,000 other neurons. For development, human DNA contains approximately 3 billion base pairs—a little under 1 gigabyte of data.

Finally, we must assume that the components of our systems are not perfect. Current fabrication techniques for silicon computers are so accurate that perfection is a reasonable assumption in many cases: errors during manufacturing can usually be detected, and the device discarded, while errors during execution are corrected using parity checks and the like. This sort of perfection is not found in biological systems: even the smallest parts will vary during development and misbehave occasionally during execution.

Using this cost model also helps insulate our designs from our rapidly changing understanding of biology, because these changes are much more likely to adjust the relative cost of parts than to entirely rule out a part.

Cost Assumptions

Based on my own highly incomplete knowledge of biology, I will propose a few assumptions about cost. While my knowledge leads me to believe these are reasonable upper bounds, it is possible that I am mistaken. If that turns out to be the case, then parts built using these assumptions will not be invalidated, but need only to have their costs adjusted.

Using these primitives requires an adjustment in how we think about building systems, thinking much more like a hardware designer than a software engineer. Once that leap has been made, however, I have found (as I begin to use them formally—I have previously used them only informally, e.g. (Beal 2002b) and (Beal 2002a)) that designing around these primitives is straightforward.

Simple Programs While the computing power of a single neuron is still unclear, a precisely constructed network of neurons can compute just about anything. For a simple program with no looping or recursion, we can assume that execution, hardware, and development costs are all proportional to the number of operations in the code plus the number of bits in the data types it manipulates.

When there is looping or recursion, then we can measure the complexity by unrolling the loops to their maximum length (unless they can be trivially parallelized) and expanding the recursions to their maximum depth.

Variation during development will cause a small percentage of programs to simply fail. Errors in execution will result in the program occasionally failing to produce a result.

Communication Paths To create a communication path between two parts, some sort of signal must be sent to guide the growing path from its source to its intended destination. Since these signals are likely to be diffusing chemicals, two signals can only be distinguished if they use different chemicals or are separated in space or time. On the other hand, a signal can guide many paths at once to the same source.

Our path creation operation is thus: given a set of source parts and a set of destination parts, create a path from every source to somewhere in the destination set.[4] The encoding cost is proportional to the number of simultaneous nearby path creation operations times the number of bits to be transmitted simultaneously on a path, and the development time is proportional to the maximum distance between source and destination parts.

Once created, a path takes hardware proportional to the number of bits that can be transmitted simultaneously. It takes a constant amount of time to send each set of bits, and

[4]Note that I place no upper limit on number of paths connected to a part, despite the fact that neurons have a fan-out limited to around 10,000 connections. This is because a part is not one, but many neurons, and the fan-out can also be boosted exponentially by adding intermediate stages (e.g. three stages is potentially 10^{12} connections).

they arrive at the other end of the path after a delay proportional to its length.

Variation during development will result in a small percentage of missing or extra paths. Errors in execution add a small amount of noise to the bits being transmitted.

Sets of Parts Filling an area with copies of a part is fairly cheap: since they develop and execute independently, both can be done completely in parallel.

During morphogenesis, coordinate systems are established with chemical gradients and used to select regions where specific development programs execute(Carroll 2005). This adds at most a constant encoding cost to the encoding cost for a single part. Since the relevant coordinate system may be established when the region is still very small, we can assume that creating a set of parts adds only a constant time cost to the time cost for developing a single part.

Once created, the hardware cost is proportional to the number of parts in the set times the complexity of a single part. The execution time for a set of parts is proportional to the execution time for a single part.

Variation during development will result in a small percentage change in the size of the set. Inclusion in a set adds no new errors to execution: the only errors are the errors of the individual parts.

Contributions

I have argued that we should evaluate the difficulty of growing devices using a biologically plausible cost metric. Like most asymptotic complexity measures, this can be used to evaluate anything from individual parts to entire systems. I have further proposed three cost assumptions for useful structures, which I have begun to use in my own work.

Developmental cost presents many important questions that must be addressed, as we adjust our view to include development. Among the most pressing:

- What are the costs of various existing architectures, such as SOAR(Wang & Laird 2006), ACT-R(Anderson *et al.* 2004) and all the rest?
- How does this view of development change the challenges facing us in system design?
- What cost assumptions are necessary, besides the three I have presented?
- Are there any circumstances where it is reasonable to *avoid* discussing development?

This proposal places our models of cognition on firmer ground, anchoring them firmly but flexibly to our understanding of both mature and developmental biology. Although the costs I have proposed may need adjustment, the analytical framework is likely to hold.

References

Anderson, J.; Bothell, D.; Byrne, M.; Douglass, S.; Lebiere, C.; and Qin, Y. 2004. An integrated theory of the mind. *Psychological Review* 111(4):1036–1060.

Beal, J. 2002a. An algorithm for bootstrapping communications. In *International Conference on Complex Systems (ICCS 2002)*.

Beal, J. 2002b. Generating communications systems through shared context. Master's thesis, MIT.

Carroll, S. B. 2005. *Endless Forms Most Beautiful*. W.W. Norton.

Wang, Y., and Laird, J. 2006. Integrating semantic memory into a cognitive architecture. Technical Report CCA-TR-2006-02, University of Michigan.