Minimal Complexity Requirements for Proteins and Other Combinatorial Recognition Systems

George D. Montañez[1][a], Laina Sanders[1][b] and Howard Deshong[1][c]
[1]*AMISTAD Lab, Department of Computer Science, Harvey Mudd College, Claremont, CA, USA*
{*gmontanez, lsanders, hcdeshong* }*@g.hmc.edu*

Keywords: Minimum Complexity, Proteins, Information Theory, Recognition Tasks, Classification

Abstract: How complex do proteins (and other multi-part recognition systems) need to be? Using an information-theoretic framework, we characterize the information costs of recognition tasks and the information capacity of combinatorial recognition systems, to determine minimum complexity requirements for systems performing such tasks. Reducing the recognition task to a finite set of binary constraints, we determine the sizes of minimal equivalent constraint sets using a form of distinguishability, and show how the representation of constraint sets as binary circuits or decision trees also results in minimum constraint set size requirements. We upper-bound the number of configurations a recognition system can distinguish between as a function of the number of parts it contains, which we use to determine the minimum number of parts needed to accomplish a given recognition task. Lastly, we apply our framework to DNA-binding proteins and derive estimates for the minimum number of amino acids needed to accomplish binding tasks of a given complexity.

1 INTRODUCTION

Complexity is costly (Ho and Pepyne, 2002; Ho et al., 2003). Ho and Pepyne, in discussing the trade-off between complexity and fragility, describe the situation this way (Ho and Pepyne, 2002):

> Consequently, as design complexity continues to increase, the probability of catastrophic bad outcomes increases. There is no avoiding this; systems become increasingly fragile as their complexity increases.

Thus, unless the extra capacity is actually needed, optimization towards robustness seems to recommend eschewing complexity in favor of simplicity. Yet, in the biological realm complexity abounds (Mitchell, 2009). In particular, molecular machines consisting of amino acid chains regularly have lengths stretching to hundreds of residues, offering an exponentially large space of possible protein structures (Branden and Tooze, 1999; Milo and Phillips, 2015). For proteins, we are moved to ask an obvious question: is all this complexity really necessary?

We propose a conditionally affirmative answer to this question, using tools from information theory and machine learning. By casting protein substrate matching as an abstract recognition task similar to binary classification (Bishop, 2006), we demonstrate that the functional capacity of proteins as information-processing recognition systems scales linearly with their size, as measured by their number of amino acids. To perform complex binding tasks, complex structures are therefore needed.

We explore the information capacity and information cost (referred to as an *information burden*) of general recognition tasks. We first define the structure of recognition tasks, including a discussion of distinguishability requirements for recognition systems relative to tasks. Representing tasks as decision trees and circuits, we provide a concrete logical structure for these abstract tasks. From this we apply a search framework to find a lower bound for the information burdens for tasks. We derive an upper bound of the recognizer's information capacity, relative to its complexity, using Abu-Mostafa and St. Jacques' method for determining the maximum information capacity of general forms of memory (Abu-Mostafa and St. Jacques, 1985). Chaining our bounds, we apply our findings to protein-DNA binding as a recognition task. We find that the minimum number of sector (functional) amino acids needed for any protein capable of performing a recognition task with information burden of b bits is linear in b, within a small

[a] https://orcid.org/0000-0002-1333-4611
[b] https://orcid.org/0000-0003-0586-4556
[c] https://orcid.org/0000-0002-3473-9239

constant factor (< 5).

2 STRUCTURE IN RECOGNITION TASKS

2.1 Defining Recognition Tasks

A **recognizer** processes a **configuration** (an object or collection of data, such as a string) and either accepts or rejects that configuration based on its features. We use the term **constraint** to signify an aspect the recognizer can test for, that must meet a certain requirement. For example, "the configuration starts with the symbol q" is one possible constraint.

Each constraint may be thought of as a logical predicate that acts on the configuration and outputs either true or false. A recognizer's behavior is expressed as a logical statement involving its constraints and the logical connectives $\wedge$ (AND), $\vee$ (OR), and $\neg$ (NOT). We require our constraints to be atomic, such that a constraint cannot be broken down into multiple simpler constraints joined with connectives. For example, "the configuration is 5 or 6 symbols long" is not one constraint; since it can be broken down at the "or" connective, it is really two separate constraints. We express each set of constraints in its minimum form (i.e., using the fewest number of connectives).

2.2 Distinguishability

Intuitively, if a recognizer treats different configurations differently, then the recognizer must be complex enough to distinguish between those configurations. Therefore, some notion of distinguishability can conceivably be used to find a lower bound on the number of constraints (and by extension, the amount of information) needed to perform a recognition task.

2.2.1 Features, Constraints, and Separability

The **features** of an object are aspects which can be in one of several possible states, or equivalently, can take values from within some set of possible values. For any given object, there are myriad ways to "featurize" it so as to arrive at some finite number of features with which to capture the important aspects of the object. As Socrates told Phaedrus, there are many ways in which to carve Nature (and we typically seek to do so at her "joints"). A common way to represent an object as a collection of features is as a vector in a high-dimensional Euclidean space, where each index of the vector corresponds to a different feature. The purpose of features in recognition tasks is to *distinguish* between objects based on their feature signatures, where a **feature signature** corresponds to the specific feature values for an object, or equivalently, refers to the specific vector in the high-dimensional feature space that represents the object.

Given a featurization of an object (i.e., representing an object as a feature signature), any two objects with the same feature signature will be indistinguishable in that feature space. This suggests that to distinguish between all objects in some set we are considering, we require a minimum collection of features, based on the number of objects we are trying to distinguish and on which distinctions matter. By the pigeonhole principle, if we have fewer possible feature signatures than we have objects, then at least two objects will share a signature and the collection of objects will not be fully distinguishable within that feature space. This will require us to add more features (or values they can take) to separate the overlapping objects.

Therefore, we say that a pair of objects is **distinguishable** with respect to a feature space if and only if they have different feature signatures in that space. As in the previous section, we can draw an equivalence between constraint sets and feature sets, considering each constraint as a binary feature. Thus, we may also speak of the distinguishability of a pair of configurations with respect to a constraint set.

For recognition tasks, we must distinguish between configurations that are accepted by the recognizer, and those that are not. This is the only distinction that matters for the task, and thus, we only require distinguishability for configurations with different labels (*accepted* or *rejected*). We say a set of configurations is **separable** with respect to a feature space if every pair of configurations in that set having different labels is distinguishable. Given a set of possible constraints viewed as binary features, we define the **minimum set of constraints** for the recognition task as the smallest subset of those features under which the set of objects under question remains separable within the smaller space.

3 REPRESENTING RECOGNIZERS

As we have seen, recognizers' behavior can be described using logical statements. We may find it convenient to re-express these statements in terms of other structures, such as decision trees or circuits, for easier analysis. We show how to do so in the next section, proving that such corresponding structures exist for all finite constraint sets.

3.1 Decision Trees

Decision trees (Quinlan, 1986) are a simple way to represent recognizers. Every non-leaf represents a constraint and has two branches off of it: one branch represents the satisfying of the constraint, and the other represents not satisfying it. Leaves represent the recognizer either accepting or rejecting a configuration. We begin reading in a configuration from the tree's root and traverse it until we reach a leaf. The process of constructing these trees from a set of constraints is outlined below, omitting details concerning the order in which you choose features via maximized information gain (as in (Quinlan, 1986)).

Theorem 1. *There exists a decision tree for every finite set of constraints.*

Proof. Recall that every set of constraints may be expressed as a logical statement. Our proof is by strong induction on the number n of logical connectives in the constraints' logical statement.

For the base case, consider $n = 0$. Note that a logical statement with 0 logical connectives represents just one constraint, and the statement evaluates true if and only if that constraint is met. We may construct a trivial decision tree for the set of constraints as follows. One node represents the constraint; if we meet the constraint, we reach an accepting leaf; otherwise we reach a rejecting leaf.

For the inductive hypothesis, we suppose that for some $k \geq 0$, every logical statement with k or fewer connectives has a corresponding decision tree.

For the inductive step, we consider the case where $n = k + 1$. Every logical statement N with $k + 1$ connectives falls into one of the cases below:

1. N is of the form $\neg A$. Since A contains k connectives, by the inductive hypothesis a decision tree exists for A. We construct a decision tree for N using the tree for A by flipping the tree's leaves: accepting leaves are now rejecting and vice-versa.
2. N consists of two logical statements A and B (respectively containing a and b connectives each, where $a+b = k$) joined by a connective. Note that A, B each have $\leq k$ connectives, so by the inductive hypothesis they have corresponding decision trees. Then N falls into a subcase below.
   (a) $N = A \wedge B$. To construct a decision tree for N, we take the decision tree for A and replace every *accepting* state with a copy of the decision tree for B.
   (b) $N = A \vee B$. To construct a decision tree for N, we take the decision tree for A and replace every *rejecting* state with a copy of the decision tree for B.

Thus, if every logical statement with k or fewer connectives has a corresponding decision tree, so does every logical statement with $k + 1$ connectives. By strong induction, we can see that every logical statement with $n \geq 0$ connectives has a corresponding decision tree. Thus, since every set of constraints is represented by a logical statement, every set of constraints has a corresponding decision tree. □

Notice that there could be many different decision trees for the same task. We would like to focus on **minimal decision trees**: decision trees with the fewest nodes needed to perform their recognition task. The size of this tree is one way to measure the minimum complexity of the recognizer.

Corollary 1.1. *There exists a minimal decision tree for every finite set of constraints.*

Proof. The proof follows directly by establishing that a decision tree exists for every constraint set by Theorem 1, then invoking the well-ordering principle on the number of nodes in the decision tree. □

3.2 Circuits

We can represent a recognizer's logical statement as a circuit in the following way (Shannon, 1938; Barnett, 2009). Let every input wire be connected to a constraint. Then following the logical statement inside out from each predicate, connect the wires through logic gates of the same type. For example, if the two predicates or statements are connected by $\wedge$ then feed both wires into an AND gate, for $\vee$ use an OR gate. Any $\neg$ in the circuit is represented by a bubble on the wire before the next gate or after the last gate to indicate the inversion.

Theorem 2. *There exists a circuit for every finite set of constraints.*

Proof. Recall that every set of constraints may be expressed as a logical statement. The result then follows immediately by strong induction on the number of logical connectives in the constraints' logical statement.

□

Corollary 2.1. *There exists a minimal circuit for every finite set of constraints.*

Proof. Use the same logic as Corollary 1.1, except with logic gates instead of nodes. □

4 INFORMATION CONSTRAINTS

By casting a recognition task in an information-theoretic search framework we can find a lower bound for the information needed to perform the task. In the same way, we can use Abu-Mostafa's formula (Abu-Mostafa and St. Jacques, 1985) to find an upper bound on the information capacity of a recognizer based on its size. Finally, we use these two results to present an inequality from which we can determine the minimum number of parts a recognizer would need to perform the task.

4.1 Information Burden

Every recognizer must have enough memory to store its constraint set—this is its **information burden**. We discuss a technique for finding a lower bound on a recognizer's information burden by reframing the task as a search problem.

Consider an arbitrary finite constraint set C. It is possible that other constraint sets exist besides C that categorize items in the exact same way. All such constraint sets fall in the equivalence class C_{eq}.

Following the discussion in Section 2.2 and using arguments similar to those from Corollaries 1.1 and 2.1, it is clear that every constraint set C has at least one minimal representation, C_{min}. Note that C_{min} does not have to be unique, but all possible selections of C_{min} must be of the same size $|C_{min}| = d$.

We approach the problem of finding the information cost of C_{min} by using the algorithmic search framework (Montañez, 2017). Let S be the set of all possible constraint sets. The collection $R_d \subseteq S$ is then all constraint sets in S with size d. Thus $C_{eq,d} \subseteq R_d$, where $C_{eq,d}$ contains all constraint sets in C_{eq} of size d.

Assuming we know d, we can determine how much information the recognizer would need to locate an element of $C_{eq,d}$ from the space R_d, which gives us a lower bound on the amount of information needed to find an element of $C_{eq,d}$ within the larger space S. In the language of algorithmic search, R_d is our search space and $C_{eq,d}$ is our target set. Assuming a search space partitioned into sets of size $|C_{eq,d}|$ and no prior knowledge of target locations, the number of bits needed to identify the target set among the partitions is

$$-\log_2 \frac{|C_{eq,d}|}{|R_d|} \qquad (1)$$

which is equivalent to the minimum number of yes/no (binary) questions one would need to ask to reduce the search space to a set containing only the target elements, under those assumptions.

Some will recognize the quantity in (1) as the Shannon surprisal of a random outcome belonging to the target set under a uniform distribution on the search space, and may be tempted to protest that this does not take into account cases where the elements of the search space are not equally likely. Indeed, when target elements are very likely, they require fewer bits of information to identify (Shannon, 1948). However, the restriction of side-information and prior knowledge prohibits us from adopting anything other than a maximum entropy (maximum uncertainty, or zero-knowledge) uniform distribution to represent our uncertainty concerning target locations (Jaynes, 2003; Dembski and Marks, 2009). Furthermore, recent results in machine learning (Montañez, 2017; Montanez et al., 2019) show that any distribution which makes target elements more likely (thus reducing the information cost of locating them) must itself be proportionally unlikely, incurring its own information burden that must be added to the total. Specifically, finding a distribution (called a *strategy*) which reduces the information cost by b' bits requires at least b' bits, so nothing is gained by positing such a favorably biasing distribution once its own information cost is taken into account (Montañez, 2017). Thus, the Shannon surprisal absent of side-information gives a lower bound on the actual information burden b, satisfying

$$-\log_2 \frac{|C_{eq,d}|}{|R_d|} \leq b$$

where b is the true **information burden** in bits.

4.2 Information Capacity

In broad terms, we may also think of a recognizer as a collection of parts placed together as some structure, some or all of which performs the recognition task.

Let M represent the number of distinct parts in some nontrivial recognizer. Then, let p represent the proportion of those parts that are actually involved in the recognition task. (Note that $0 < p \leq 1$, with an exclusive lower bound since at least one part is involved in nontrivial recognition.) We can see that the number of parts involved in recognition is $Mp = n$.

Now, if we let k denote how many types of parts can be arranged to form a recognizer, then we can see that k^n is an upper bound on the total number of recognizers that use n parts for the recognition task, since there are at most k choices for each of its n parts. If we consider these k^n recognizers as a whole, they demonstrate at most k^n distinct behaviors (that is, they accept/reject configurations according to distinct criteria).

Now recall from Abu-Mostafa (Abu-Mostafa and St. Jacques, 1985) that the information capacity of a structure that can distinguish between d items is $\log_2(d)$. The information capacity c of our space of k^n recognizers satisfies

$$c \le \log_2(k^n) \in O(n)$$

since the collection can distinguish between at most k^n behaviors.

Since the information burden must be less than the capacity, we know that if $b > f(n)$ for all functions $f(n)$ linear in n, then our recognizer does not have enough parts or capacity to perform the task. Conversely, assuming it can perform the task requires that its capacity is no less than the information burden for that task, namely, that $b \le c$.

Chaining our bounds, we obtain

$$-\log_2 \frac{|C_{eq,d}|}{|R_d|} \le b \le c \le \log_2(k^n) \qquad (2)$$

or more simply

$$-\log_2 \frac{|C_{eq,d}|}{|R_d|} \le n \log_2 k. \qquad (3)$$

5 PROTEINS AS RECOGNIZERS

Biologists believe there are at least tens of thousands of different proteins in the human body (Ponomarenko et al., 2016), each with a specific set of tasks (Branden and Tooze, 1999). Some of these tasks involve the protein binding to another molecule, such as an enzyme binding to substrate. For restriction enzymes and some other families of proteins, the substrate molecule is DNA. In these cases, the enzymes have a distinct set of DNA sequences that can be bound based upon affinities between amino acids in the enzyme and the base pairs within the DNA.

Let us consider proteins binding to a specific set of DNA sequences as a recognition task. In this scenario, the protein is the recognizer while DNA sequences are the configurations. We can determine the information burden and capacity of the abstracted view of a protein, using the bounds established earlier. Since there are 20 frequent amino acids, let $k = 20$. The lower bound term is specific to the protein's task, so we cannot simplify it any further. Using Inequality (3) we then obtain,

$$-\log_2 \frac{|C_{eq,d}|}{|R_d|} \le n \log_2 20 < 5n.$$

Thus we can determine the minimum number of sector (functional) amino acids (McLaughlin Jr et al., 2012), denoted n, required for a protein to perform a specific task (i.e., bind to specific subset of all DNA sequences) as

$$n > -\frac{1}{5} \log_2 \frac{|C_{eq,d}|}{|R_d|}$$

which, given binary predicates as constraints (implying $|R_d| \ge 2^d$), can be rewritten as

$$n > \frac{1}{5}\left(d - \log_2 |C_{eq,d}|\right). \qquad (4)$$

In the case that the minimum constraint set is unique (namely, $|C_{eq,d}| = 1$), then we obtain the simplified

$$n > \frac{1}{5} d, \qquad (5)$$

where d is the number of constraints in the minimal set.

5.1 Discussion

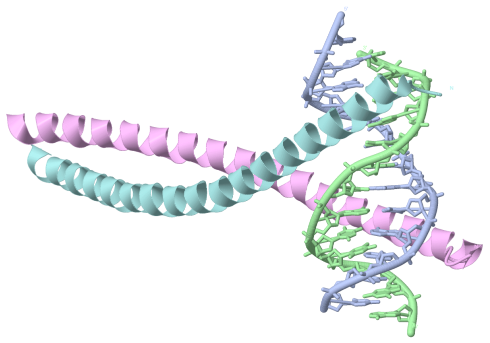

Figure 1: [Example] Crystal structure of leucine zipper bzip transcription factor PAP1 bound to DNA, requiring multi-turn alpha helix components. (PDBid 1GD2/PD0180, (Fujii et al., 2000))

Inequality (4) gives a lower bound for protein recognition systems based on a completely abstract view of proteins as combinatorial recognition systems. Given that the median protein length is roughly 300 residues (Brocchieri and Karlin, 2005), and assuming that roughly 20% of the residues are sector residues (McLaughlin Jr et al., 2012), this bound tells us that proteins would be able to accomplish recognition tasks requiring up to 300 constraints, which appears to be a generous overestimate. Keeping a completely abstract view of recognition systems appears to come at a cost, but not one without remedy. We can tighten the bound by putting back into our model some of the specifics concerning DNA-binding proteins.

Our initial overestimation is partially due to the abstract model allowing for cases where the choice of each amino acid transmits $\log_2(20) \approx 4.32$ bits of information, potentially allowing a single sector residue to process and determine the outcome of four distinct binary constraints. Real proteins, however, regularly require secondary structure in their constraint handling, such as alpha helices and beta sheets, which are substructures within proteins formed by the three-dimensional arrangement of multiple amino acids (Pauling et al., 1951; Pauling and Corey, 1951). For example, DNA-binding domains such as leucine zippers, helix-turn-helix and winged-helix structures require multiple alpha helices (Korasick and Jez, 2015; Fujii et al., 2000), which themselves require 3.6 amino acids per turn. Thus, an alpha helix with 10 turns requires 36 amino acids, and if just two alpha helices of this size are needed to process a specific constraint, this increases the residue count by a factor of 72. Rather than a single amino acid handling many constraints, it is more typical that a single constraint requires multiple amino acids working together to process it.

Plugging such biology-specific factors into the model will yield tighter lower bounds on residue counts. We can add a protein-specific scaling factor α which for each sector residue required by the general model will add a multiplicative factor reflecting the biological need for secondary support structures, taking into account the additional amino acids each sector residue carries with it to fulfill its role. The modified bound then becomes

$$n > \frac{\alpha}{5}\left(d - \log_2 |C_{eq,d}|\right) \tag{6}$$

with corresponding simplification

$$n > \frac{\alpha}{5} d \tag{7}$$

when C_{min} is unique.

As an example, if we again assume two alpha helices of ten turns are required for each constraint, this gives a scaling factor of $\alpha = 72$. By Inequality (7), a protein testing four constraints will require at least 57 sector residues, which corresponds to a protein of 285 total residues (assuming sector amino acids account for around 20% of all residues in a typical protein). This biology-specific estimate agrees with known median protein lengths for eukaryotes and prokaryotes (Brocchieri and Karlin, 2005), suggesting the utility of attaching domain-specific considerations to our general model, while still allowing us to abstract away many of the intricacies of protein-DNA interactions such as how proteins deform DNA structure or how specific proteins and DNA sequences do not always interact in the same way.

6 ALTERNATIVE MODELS OF PROTEIN-DNA INTERACTION

Just as using biology-specific details can help improve analysis, one might be tempted to consider using other, less-abstract, models of computation to analyze the minimal complexity requirements for DNA-binding recognition tasks. Such an idea is not without merit, but we find that if we use other common models of computation difficulties quickly arise. While they can model the specific functions of DNA-binding proteins in some ways, they suffer as we try to encompass additional aspects of the interactions. We outline two such models with a discussion of their application and shortcomings below.

6.1 Deterministic Finite Automaton

Following the standard construction (Sipser et al., 2006), a **deterministic finite automaton** (DFA) is a five-tuple $(Q, \Sigma, \delta, q_0, F)$. Here, Q is a set of states and $q_0 \in Q$ is the starting state; Σ is a set of symbols that may be strung together as inputs to the DFA; the DFA reads in its input left-to-right one symbol at a time and moves between states according to the transition function $\delta : Q \times \Sigma \to Q$; and $F \subseteq Q$ is the set of accepting states.

6.1.1 Shortcomings

The DFA model fails to capture the way that DNA-binding proteins read in their input. The model reads its input left-to-right in one direction. In actuality, proteins move probabilistically along DNA as they read it; at any point, protein may move left, move right, jump along the DNA, or even disconnect entirely from the DNA to end computation prematurely (Stanford et al., 2000).

6.2 Sliding Window Protein Automaton

A **sliding window protein automaton** (SWPA) is a tuple $(Q, \Sigma, \delta, A, P, w)$. Recalling other automaton constructions, Q is a set of states; $A \subseteq Q$ is the set of accepting states; and the alphabet Σ is a set of symbols. The automaton reads a finite tape containing symbols in Σ using a sliding window that views a continuous block of w symbols at a time. The window begins at a random spot on the tape. $P = \{p_l, p_r, p_x\}$ contains the probabilities that, at each step of the automaton's run, the automaton's window will slide one symbol left, one symbol right, or fall off. (If the automaton reaches the right end of the tape and slides further right, or vice-versa, it simply falls off.) Note that $p_l + p_r + p_x = 1$. Finally, the transition function $\delta : Q \times \{L,R\} \times \Sigma \to Q$ describes the state to which the automaton moves at each step of the computation, given the direction its window slides and which new symbol enters the window as it slides. If the protein ever enters an accepting state, then the computation stops and accepts. Otherwise the computation runs until the window falls off.

6.2.1 SWPAs and Distinguishability

The notion of language distinguishability can be used to find a lower bound on the number of states in a DFA. We discuss why the traditional distinguishability formulation is not well-suited for SWPAs and provide a new notion of distinguishability more relevant to SWPAs.

Traditionally, two strings a,b are distinguishable with respect to a language L if there exists some (potentially empty) string z such that L contains az but not bz, or vice-versa. Intuitively, this formulation is useful because if a and b are distinguishable with respect to L, then any DFA accepting L must reach a different state when it reads a as opposed to b. Thus, distinguishability lets us establish a lower bound on the number of states in L.

Distinguishability relies on the assumption that our automaton reads in strings without ever changing its read direction. Since this assumption holds for DFAs, distinguishability is a powerful tool for DFA analysis; however, the assumption does not hold for SWPAs, so this form of distinguishability is not useful for that context. Intuitively, if az is in L while bz is not, then just knowing that a string ends with z is not enough to say whether or not it is in L. So, as a DFA for L reads in the string az moving left to right, the DFA must somehow “remember” that the string began with a instead of b, but it cannot read the string in reverse to check. The DFA “remembers” by encoding a and b in different states. However, SWPAs do not read in strings in one direction: at every step of computation, their sliding window may move left or right, and this movement is probabilistic. Thus, we cannot distinguish two strings by appending a common suffix to them both—in general, we cannot guarantee that the SWPA will read the suffix! Since SWPAs do not read strings in one direction, we must reformulate our notion of distinguishability to apply it to SWPAs.

For an SWPA, two strings a,b are distinguishable if there exists a list of SWPA instructions $i \in \{(d,\sigma) \mid d \in \{l,r\}, \sigma \in \Sigma\}$ such that, if we begin with our window containing a and perform i in order, we end up in an accepting state, whereas if we perform the same process starting with our window containing b, we end up in a rejecting state (or vice-versa).

6.2.2 Shortcomings

This model aims to capture information about the complexity of proteins’ binding sites. However, in its present form, it fails to encode information available to the binding site, and it acts upon information that the binding site does not have access to.

The model is capable of storing information that the binding site cannot. In general, after a protein’s binding site tests and rejects a segment of DNA, the protein does not store a “memory” of this rejection. On the other hand, memory is a central aspect of our SWPA model. Whenever the SWPA slides along its tape, it undergoes a state transition. Unless every state in the automaton has a transition to every other state, and those transitions are all be taken regardless of the sliding window’s current contents (which, by SWPAs’ determinism, cannot hold unless SWPAs only have one state), then the automaton’s present state will always contain some information about where and/or in what order the sliding window has been on the tape previously. That is, the SWPA always has access to more “memory” than the binding site unless the SWPA only contains one state—in which case it either accepts every string (if the state is accepting) or rejects every string. It seems, then, that although the SWPA model captures the way proteins move between steps of computation, it does not suitably describe how the proteins perform each step of the computation.

7 CONCLUSION

Our work here concerns recognition problems, in which recognizers evaluate configurations according to constraints. The constraints are equivalent to sets of logical statements, and as such, recognition problems can also naturally be framed in terms of structures such as decision trees or circuits. Using information-theoretic results related to search and general memory capacity, we derived bounds on the complexity of recognizers, bounds which limit the complexity of tasks they can perform. The information capacity of recognizers must at least equal the information burden of a task for the recognizer to be able to perform it. Thus, we can use the information burden to lower-bound the combinatorial complexity of an abstract recognition system, if we know that the task is performed by the system in question. Finally, we suggest DNA-binding as a simple biological application of our framework, and explain how other computational models fail in various ways for this particular application.

ACKNOWLEDGEMENTS

This work was financially supported by the NSF under Grant No. 1659805, Harvey Mudd College, the Rose Hills Foundation, and by a grant from the Walter Bradley Center for Natural and Artificial Intelligence.

We would like to thank Karl Haushalter, Steve Adolph, Eliot C. Bush, Jae Hur, Douglas Axe, Joe Deweese, and Matina Donaldson-Matasci for their insights and suggestions on early iterations of this work.

REFERENCES

Abu-Mostafa, Y. and St. Jacques, J. (1985). Information capacity of the Hopfield model. *Information Theory, IEEE Transactions on*, 31(4):461–464.

Barnett, J. H. (2009). Applications of Boolean Algebra: Claude Shannon and Circuit Design. *Available from the webpage http://www.cs.nmsu.edu/historical-projects*.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Branden, C. I. and Tooze, J. (1999). *Introduction to Protein Structure*. Garland Science. 2nd Edition.

Brocchieri, L. and Karlin, S. (2005). Protein length in eukaryotic and prokaryotic proteomes. *Nucleic acids research*, 33(10):3390–3400.

Dembski, W. A. and Marks, R. J. (2009). Bernoulli's principle of insufficient reason and conservation of information in computer search. In *2009 IEEE International Conference on Systems, Man and Cybernetics*, pages 2647–2652. IEEE.

Fujii, Y., Shimizu, T., Toda, T., Yanagida, M., and Hakoshima, T. (2000). Structural basis for the diversity of dna recognition by bzip transcription factors. *Nature Structural & Molecular Biology*, 7(10):889.

Ho, Y.-C. and Pepyne, D. L. (2002). Simple explanation of the no-free-lunch theorem and its implications. *Journal of optimization theory and applications*, 115(3):549–570.

Ho, Y.-C., Zhao, Q.-C., and Pepyne, D. L. (2003). The No Free Lunch theorems: Complexity and Security. *IEEE Transactions on Automatic Control*, 48(5):783–793.

Jaynes, E. T. (2003). *Probability theory: The logic of science*. Cambridge university press.

Korasick, D. and Jez, J. (2015). Protein domains: Structure, function, and methods. In Bradshaw, R. A. and Stahl, P. D., editors, *Encyclopedia of cell biology*. Academic Press.

McLaughlin Jr, R. N., Poelwijk, F. J., Raman, A., Gosal, W. S., and Ranganathan, R. (2012). The spatial architecture of protein function and adaptation. *Nature*, 491(7422):138.

Milo, R. and Phillips, R. (2015). *Cell Biology by the Numbers*. Garland Science.

Mitchell, M. (2009). *Complexity: A Guided Tour*. Oxford University Press.

Montañez, G. D. (2017). The Famine of Forte: Few Search Problems Greatly Favor Your Algorithm. In *Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference on*, pages 477–482. IEEE.

Montañez, G. D. (2017). *Why Machine Learning Works*. PhD thesis, Carnegie Mellon University.

Montanez, G. D., Hayase, J., Lauw, J., Macias, D., Trikha, A., and Vendemiatti, J. (2019). The Futility of Bias-Free Learning and Search. *arXiv preprint arXiv:1907.06010*.

Pauling, L. and Corey, R. B. (1951). Atomic coordinates and structure factors for two helical configurations of polypeptide chains. *Proceedings of the national academy of sciences of the United States of America*, 37(5):235.

Pauling, L., Corey, R. B., and Branson, H. R. (1951). The structure of proteins: two hydrogen-bonded helical configurations of the polypeptide chain. *Proceedings of the National Academy of Sciences*, 37(4):205–211.

Ponomarenko, E. A., Poverennaya, E. V., Ilgisonis, E. V., Pyatnitskiy, M. A., Kopylov, A. T., Zgoda, V. G., Lisitsa, A. V., and Archakov, A. I. (2016). The size of the human proteome: the width and depth. *International journal of analytical chemistry*, 2016.

Quinlan, J. R. (1986). Induction of Decision Trees. *Machine learning*, 1(1):81–106.

Shannon, C. E. (1938). A Symbolic Analysis of Relay and Switching Circuits. *Electrical Engineering*, 57(12):713–723.

Shannon, C. E. (1948). A Mathematical Theory of Communication. *Bell system technical journal*, 27(3):379–423.

Sipser, M. et al. (2006). *Introduction to the Theory of Computation*, volume 2. Thomson Course Technology Boston.

Stanford, N. P., Szczelkun, M. D., Marko, J. F., and Halford, S. E. (2000). One-and three-dimensional pathways for proteins to reach specific dna sites. *The EMBO Journal*, 19(23):6546–6557.