Designing More "Human-like" Algorithms: A computational complexity perspective

Megha Srivastava

March 2023

1 Introduction

We encounter and solve a variety of algorithmic tasks in our daily lives without intentionally formalizing the underlying algorithm, such as sorting through a pile of laundry, remembering long numerical sequences (e.g. social security numbers), and creating schedules under a given set of constraints. In fact, such examples often inspired well-known problems in theoretical computer science – for example, the "traveling salesman problem" (an NP-hard problem) phrase was first mentioned in a 1832 German handbook describing a 47-city tour through Germany for a salesman that does not visit any city more than once (Applegate et al., 2006). However, while we may often wish to solve such tasks optimally, humans are not always capable of analyzing and discovering the optimal solution to an algorithmic task. Instead, we tend to develop heuristics or rely on instructions from family and friends to guide our strategies, and are also faced with limitations (perceptual, memory, etc.) that either alter the overall task complexity or lead to suboptimal behavior.

A diverse set of works have attempted to discuss the links between human strategies when solving a given task and its computational complexity. Such studies explore a variety of modalities, ranging from perceptual tasks (e.g. designing compact frequency-based word clouds) to decision-making (e.g. choosing a path to travel on) to social behavior (e.g. forming friendships and alliances), and often seek to answer one of the following questions:

- When does computational complexity (as opposed to other features) serve as a strong predictive signal for human behavior?
- When (and why) does human behavior deviate from computationally optimal strategies?
- When does diversity (e.g. different solutions to the same problem) in behavior arise?

However, these works largely come from researchers who are primarily concerned with developing stronger models of human intelligence in decision-making contexts. For example, Lieder and Griffiths (2020) describe a unifying paradigm of *resource-rational analysis* that covers a variety of empirical findings that humans deviate from optimal behavior due to cognitive limitations or the use of heuristics, while Gershman et al. (2015) present a unifying view of *computational rationality*, which describes the goal of making real-world decisions with the highest expected utility under computational constraints. Many of the works covered by these paradigms focus on tasks requiring human decision making under uncertainty using probabilistic inference, which has been shown to belong to the NP-hard complexity class (over a general class of Bayesian belief networks) (Cooper, 1990). But what about other types of tasks, that more closely map to some of the classical problems studied in introductory computational complexity?

We provide an accessible overview to classical complexity problems with known-optimal algorithms (and accompanying proofs), accompanied by corresponding empirical results that explore human behavior when solving such problems in real-world contexts. Why should one care about such connections? One practical reason is that by understanding the general resource constraints that affect human behavior, we can hope to develop stronger algorithms for assistive technologies (e.g. navigation tools or virtual chess tutors [1]) that compensate for human sub-optimalities (e.g. inability to perform exhaustive search). However, our overall goal is to encourage people who may have little familiarity with either computational complexity theory or cognitive science (or both) to better reason about the way complexity intersects our daily lives through simple examples.

2 Travelling Salesman

The Travelling Salesman Problem is one of the most famous problems in optimization. The goal is to find the route with shortest cost (e.g. length) between a set of points (e.g. locations) that must be visited, and underlies many transportation problems.

[1] https://noctie.ai/

2.1 Overview

Let us consider a graph G over a set of n vertices V connected via weighted edges E. We define a Hamiltonian cycle to be a cycle through this graph that visits every vertex $v_i \in V$ exactly once. The Travelling Salesman Problem (TSP) essentially asks us to find the Hamiltonian cycle with the smallest overall cost (sum of all edge weights). While the decision version of the problem (deciding if there exists a given graph G with a Hamiltonian cycle of cost $\leq k$) is NP-complete, the optimization problem as described above is only known to be NP-Hard. Furthermore, we can also consider a Euclidean variant of the problem, where each vertex $v \in V$ is a point in R^2 space, and the weight of edge $e_{i,j}$ between v_i and v_j is equal to the Euclidean distance between both points, $||v_i - v_j||_2$.

2.2 Exact Algorithm

One natural algorithm to solve the Travelling Salesman Problem is a brute force approach, where we search between all possible Hamiltonian cycles (or permutation of vertices in V), and pick the one with the smallest cost. This would require searching across $n!$ paths, which is not very efficient. Here, we describe the Held-Karp algorithm, which finds an exact solution for TSP using dynamic programming.

1 (Held-Karp Algorithm) *:*
Select a starting vertex $s \in V$. For every subset $S \subset V - s$ of vertices not including s, and vertex $v' \in S$, let $d(S, v')$ be the cost of the lowest cost path from s to v' that also visits every vertex in S.

Base case: $S = v', d(S, v') = c(s, v')$ where c is the cost function over edge $e_{s,v'}$.

For S when $|S| \geq 2$, we follow the recursive procedure:
$d(S, v') = min\{d(S - v', u) + c(u, v') | u \in S - v'\}$

Calculate the lowest cost path as $min\{d(V - s, v') + c(s, v') | v' \in V - s\}$.

Because the number of subsets of a set of size n is 2^n, the overall time complexity of the Held-Karp Algorithm is $\theta(2^n * n^2)$, which is better than the brute force approach wtih complexity O($n!$).

2.3 Approximate Algorithm

An efficient algorithm to solve TSP approximately is the Nearest Neighbors algorithm, which greedily visits the nearest point, as described below.

2 (Nearest Neighbors) *:*

Mark all vertices in V as "unvisited". Select a starting vertex $s \in V$, and mark it as "visited".

For all $v_i \in V$ where v_i is unvisited, calculate the cost $c(s, v_i)$, and identify the vertex v_i' with the smallest cost. Reset the starting vertex s to be v_i', and mark v_i' as "visited".

Repeat the above procedure until all $v_i \in V$ are marked as "visited".

Because we now need to calculate the distance between a vertex on all remaining unvisited vertices, the time complexity is $\theta(n^2)$.

2.4 Human behavior

For both exact (e.g. Held-Karp) and approximate (e.g. Nearest Neighbors) algorithms, we analyze complexity in terms of time complexity, which in turn is a function of the number of vertices n. However, a key principle behind understanding more "human-like" approaches to such problems is considering what other measures of complexity drive human behavior. Consider the Euclidean variant of TSP mentioned earlier. It has been shown that the optimal path for any set of vertices, solving the TSP, in the Euclidean plane does not contain "self-intersection" – essentially, no two paths cross (Flood, 1956). Guided by this knowledge, one can essentially consider the *convex hull* of the set of vertices, and make sure to connect all vertices on the boundary sequentially, as shown in Figure 1. This largely leaves uncertainty on how to visit the "interior points", and so we can consider the number of interior points as a notion of complexity instead (MacGregor and Omerod, 1996).

This is in fact what MacGregor and Omerod (1996) study in two human subject studies, where they find that as the number of interior points increases, human subjects identify significantly more

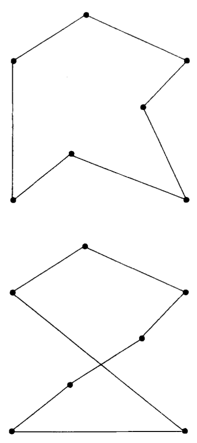

Figure 1: Two possible paths between vertices in the Euclidean plane (MacGregor and Omerod, 1996). The above path has no intersecting paths, all the boundary vertices are connected sequentially, and is optimal for the travelling salesman problem defined over these vertices. However, the bottom (non-optimal) strategy shows an example of intersecting paths and not following consecutive vertices along the convex hull.

optimal solutions than heuristics such as Nearest Neighbors described earlier (e.g. 3.8 vs. 10.4 average percent above optimal path length). Furthermore, they find that solutions had significantly less number of indentations than the expected value, suggesting that human subjects were indeed guided by the global properties of the convex hull. However, while MacGregor and Omerod (1996) find no notable difference between participants, Vickers et al. (2004) present alternate findings when increasing the *path complexity* of different TSP instances, which measures the degree at which vertices should be connected to their nearest neighbor in the optimal path. They find that for sufficiently complex problems, human performance correlated with their performance on the Raven's Advanced Progressive Matrices test, a cognitive exam that assesses non-verbal skills, and propose using problems like TSP. where there may be multiple optimal solutions. as important cases for study human problem-solving abilities (Vickers et al., 2004).

One reason why understanding what forms of complexity of the TSP influence behavior can be useful is, as suggested in the introduction, for designing assistive technologies for humans. Krolak et al. (1970) demonstrate this by framing TSP, which itself underlies common tasks like circuit board wiring, as a human-computer cooperative task where a computer and human iterate on refining initial solutions. Their approach relies on the assumption that humans, who have the ability to visualize the entire set of vertices, are particularly skilled at identifying global, visual properties, and can therefore rely on heuristics such as a good solution to TSP likely having less indentations and is more polygonal in nature. Using vertices that are locations selected from a map of the United States, Krolak et al. (1970) show that study participants are able to arrive at solutions close to one percent in length of the optimal solution.

3 Vehicle Routing

The Vehicle Routing Problem (VRP) is a generalization of the Travelling Salesman Probelm (TSP), where the goal is to find an optimal *set* of routes for *multiple* vehicles instead just one, which would be equivalent to TSP. There are many different factors underlying optimality, including the total overall cost of all routes, the maximum cost of single vehicle's route, and vehicle capacity constraints. Any approach to solving the vehicle routing problem naturally fall into two steps: first, assigning different locations (points) to each vehicle, and second, deciding the route for each vehicle to follow.

3.1 Overview

Let us define a transportation graph G over a set of n vertices V. Let v_0 denote a special "depot" vertex, from which an unlimited fleet of vehicles can travel from, K represent the capacity of each vehicle, and $q_0 ... q_n$ represent the weight demand of each vertex in V. The goal is to minimize the total distance travelled by all vehicles in the fleet, where distance $d(v_9, v_j)$ is measured using a distance metric d such as the Euclidean distance. Since the case of 1 vehicle reduces to a TSP instance, it is clear that this VRP problem is NP-hard, and we will primarily focus on describing a common heuristic.

3.2 Savings Heuristic

A common heuristic approach to VRP is the "savings" algorithm proposed by Clarke and Wright (1964). The overall approach is to consider the savings in total distance a single vehicle visits two vertices on the same route, versus two vehicles separately visiting each vertex.

3 (Clark-Wright Savings Algorithm) *:*
Let $s(v_i, v_j) = 2d(v_0, v_i) + 2d(v_0, v_j) - (d(v_0, v_i) + d(v_i, v_j) + d(v_0, v_j)) = d(v_0, v_i) + d(v_0, v_j) - d(v_i, v_j)$*, representing the savings when a single vehicle visits* v_i *and* v_j*.*

Calculate $s(v_i, v_j)$ *across all pairs* $v_i \in V, v_j \in V$*, and let* L *be a sorted list of all* $s(v_i, v_j)$ *in descending order.*

Initialize a starting current route r *containing the first* $s(v_i, v_j)$ *in* L*. For each remaining* $s(v_i, v_j)$ *in* L*, if neither* v_i *nor* v_j *are in* r*, initialize a new route with that pair.*
If v_i *an* v_j *exist in two different routes, merge the two routs as loss as the capacity constraint is met (sum of* q_i *for each* v_i *in route* r *is less than K).*
If either v_i *or* v_j *exist in a, add* (v_i, v_j) *to the route as loss as the capacity constraint is met (sum of* q_i *for each* v_i *in route* r *is less than K).*

Once the list is processed, return all routes.

The time complexity of this algorithm is $O(n^2 log_2(n))$.

3.3 Human behavior

Recent work has sought to model human strategies when solving VRP, and compare these strategies to three heuristics, including the savings heuristic, across VRP problem instances with a variety of topology (e.g. number of nodes, local structures) Fontaine et al. (2020). As the Clark-Wright savings algorithm is a global heuristic strategy, it is strong when relying on local structure first (e.g. first visually clustering vertices before creating routes) is not optimal. Interestingly, Fontaine et al. (2020) do find that human participants modify their strategy depending on problem instances, and in general there is more variance in solutions between participants in the multiple vehicle setting (vs. TSP). The average human performance is between the first and second best heuristic strategy for each instance, and similar to Vickers et al. (2004), there is correlation between aptitude test performance (in this case, the Cognitive Reflection Test) and overall optimality of solutions. Interestingly, as human participants with low Cognitive Reflection Test performance tended to create clusters (i.e. assignment of vertices to different vehicles) for which the TSP problem is easier to solve, while those with high Cognitive Reflection Test performance invested in creating good clusters that led to stronger overall solutions. Finally, Fontaine et al. (2020) find that providing feedback (in the form of giving the overall distance of the current solution) helps can generally significantly help improve performance, but for subjects with low Cognitive Reflection Test actually *disimproves* performance, which is hypothesized to be due to an overreaction towards feedback.

4 Exact Cover

The Exact Cover Problem is a type of constraint satisfaction problem, where the goal is to find a collection of subsets that contain every element of a set exactly once. It underlies many popular puzzles, including Sudoku, finding Latin squares, and Pentomino tiling puzzles.

4.1 Overview

Let A be a binary matrix where each cell is either 1 or 0. The goal of the Exact Cover Problem is to identify a subset of the rows of A such that the digit 1 appears in each column exactly once. Intuitively, each column represents a "constraint", and the overall problem is NP-Complete (via an equivalence to 3-SAT). We can see that the common puzzle Sudoku maps to this, where a particular number can must be used exactly once in a given row, column, and block.

4.2 Exact Algorithm

One of the most well-known exact algorithms for the Exact Cover Problem is the recursive Algorithm X, described by Knuth (2000) to show the power of "dancing links" as an efficient way to implement backtracking.

4 (Algorithm X) *:*
Let A *be the input binary-valued matrix, which we will modify and return. Initialize a solution set* S *which will contain the rows in the solution.*

If A *is empty, terminate return the current solution set* S*.*

Otherwise, pick a column c, and a random row r such that $A[r][c] = 1$. Include r in the current partial solution set S.
For every column j where $A[r][j] = 1$, delete every row i where $A[i][j] = 1$ and then delete column j from A.

Repeat the above algorithm recursively until it terminates.

The algorithm essential performs a depth-first search, where the original matrix A is the root of the search tree, and has exponential running time.

4.3 Human behavior

Search problems in general are challenging for humans due to our limited memory – to perform an algorithm requiring backtracking, a human would need to remember a large number of candidate solutions at the start. Therefore, humans tend to rely on logical heuristics to aid their search, which often deviate far from the optimal strategy. In Sudoku, popular logical strategies range from identifying “naked pairs”, where two cells in the same row, column, or block share the same possibilities of two candidates, to complex strategies for eliminations such as the *Finned Mutant Swordfish*, which requires constraints on sets of three rows, columns, or blocks (Stuart, 2007). Moreover, Chapman and Rupert (2012) propose a group-theoretic approach for representing human Sudoku-solving strategies, leveraging the fact that humans tend to first identify constraints on a cell (e.g. numbers it cannot be assigned to), and that two boards are similar if they can be transformed to each other without introducing new constraints. Likewise, Lee et al. (2009) conduct a series of interesting human subject experiments that show that indeed, novice humans are more likely to rely on simple “exclusion tactics”, focusing on what numbers a cell cannot be assigned, than “inclusion tactics”. They define the “ relational complexity” of a tactic as the number of constraints needed to determine the target variable, or in the context of Sudoku, the number of digits to consider, and found a significant increase in latency of human participants as the complexity of required tactics increased (Lee et al., 2009), showing that the number of constraints and “options” to keep in mind is a strong influence on human behavior.

5 Knapsack

The Knapsack Problem is a well-known combinatorial optimization problem inspired by the resource allocation problem faced when one needs to choose which items to fill a fixed-size knapsack with. It underlies many real-world applications, from knapsack-based encryption systems to home energy and power management.

5.1 Overview

Let there be n items, number 1 to n, which each have a weight w_i and value v_i. Furthermore, let W represent the maximum weight capacity of a knapsack. Let x_i be the number of copies of each item to place in the knapsack. The goal of the Knapsack Problem is to *maximize* $\sum_{x=1}^{n} v_i x_i$ subject to the constraint $\sum_{i=1}^{n} w_i x_i \leq W$. For the 0-1 Knapsack Problem, we add the constraint that $x_i \in 0, 1$. Although the decision version of the problem (whether an assignment of items that achieve a certain value V subject to weight constraints exists) is NP-complete, the actual optimization problem is NP-hard.

5.2 Exact Algorithm

While the brute force approach for solving the Knapsack Problem would be to try all 2^n possible subsets of n, a more efficient method using Dynamic Programming, as shown below:

5 (Dynamic Programming) *:*
Define a matrix M such that $M[i, w]$ stores the maximum value that we can achieve using all items up to item i, and with weight less than or equal to w.

Initialize $M[0, w] = 0$.
If $w_i > w$, then we should not add the a new item i, so $M[i, w] = M[i - 1, w]$.
Otherwise, if $w_i < w$, then $M[i, w] = max(M[i - 1, w], m[i - 1, w - w_i] + v_i)$.

Starting from $i = 0, w = 0$, use dynamic programming with the above rules to fill all entries of M.
Return $M[n, W]$.

The overall time complexity of the algorithm is $O(nW)$, which is *pseudo-polynomial*, meaning it is polynomial with respect to the value of W, but not its length.

5.3 Approximate Algorithm

As discussed earlier, algorithms that require storing lots of sub-solutions in memory (like dynamic programming approaches) are often challenging for humans to employ as strategies. A very intuitive approach is the greedy algorithm, where one greedily selects items with the most value, which does not solve the Knapsack Problem exactly. A stronger approximation algorithm proposed by Sahni (1975) is shown below:

6 (Sahni k-approximation) *:*
Given integer input k, let I be the set of all subsets s of n items such that $|s| = k$ and $\sum_{j \in s} w_j < W$.

Initialize $P_{MAX} = \{\}$.
For each s in I, set $P = s$. Then, greedily add remaining items into P until $\sum_{j \in P} w_j \geq W$.
If $\sum_{j \in P} v_j \geq \sum_{j \in P_{MAX}} v_j$, set $P = P_{MAX}$.

After iterating through every s in I, return P_{MAX}.

Intuitively, this algorithm improves upon greedy by allowing us to explore adding items the greedy approach would not identify via the input k. The runtime of this algorithm is $O(n^k)$, and we can define a notion of *Sahni difficulty level k* as the number k at which a given Knapsack Problem instance can be solved exactly, but for $k - 1$ cannot be.

5.4 Human Behavior

Many cognitive assesment tests use tasks that reduce to the Knapsack Problem, such as The Hotel Task on which low-functioning ADHD patients have been found to perform significantly worse on (Torralva et al., 2013). This warrants a closer look at what strategies humans employ when solving such problems. The *Sahni difficulty level k* described above is what Murawski and Bossaerts (2016) select as a measure of problem complexity to study human behavior when solving eight different instances of the 0-1 Knapsack Problem. Participants in their human subject study were provided a computer interface to assign items to a digital knapsack, and Murawski and Bossaerts (2016) showed that as the *Sahni difficulty level k* increased, the overall "economic performance", or closeness in total value with the value of an optimal solution, decreased, suggesting a strong correlation between a measure of computational complexity and human performance. This supports the hypothesis that a greedy approach is intuitive for human behavior, and the closer a Knapsack Problem instance is for a greedy approach to be successful, the stronger human performance will be. Interestingly, Murawski and Bossaerts (2016) also found that more than 30% of the time, participants were unable to recover the same solution when provided the same instance again, suggesting that they were not aware of the optimality of their approach. Furthermore, there was strong heterogeneity amongst solutions, showing that humans exhibit diversity even when there is a fundamental structure behind their approach.

6 Numerical Processing & Basic Arithmetic

Beyond classical problems in theoretical computer science, we can also consider complexity of more simple tasks, such as basic arithmetic operations and numerical processing. In cognitive science, numerical processing refers to how humans represent different numbers, which is influenced not only by biological development, but also by cultural and social factors such as early mathematics education .

6.1 Overview

When analyzing the computational complexity of the simple arithmetic operations we perform in our daily lives, such as subtraction and multiplication, we focus on the computational binary representation of numbers. Concretely, a number N is represented as a sequence of $n = floor(logN) + 1$ bits (0 or 1), and arithmetic operations are carried out on these sequences by bitwise operators such as "AND". The time complexity of arithmetic operations ultimately relies on the operators needed for processing the binary-representation of numbers:

- **Addition/Subtraction**: $\theta(n)$
- **Multiplication**: $O(n^2)$ using the standard "grade school" algorithm, or $O(n^{1.585})$ using Karatsuba's algorithm, which is optimized for bit operations on a computer and reduces standard multiplication into three smaller multiplications (Karatsuba and Ofman, 1962)

- **Parity**: $O(1)$, by just checking the last bit

6.2 Human behavior

For computers, the complexity of simple arithmetic operations is driven by the fact that we represent numbers as bits – e.g., regardless of their values, subtracting two $n - digit$ numbers always has time complexity $\theta(n)$. However, research on numerical processing has shown that humans represent numbers very differently. For example, discriminating (e.g. identifying two values are different) between 16 and 32 is much easier for infants that discriminating between 16 and 24, and the "precision" of such numerical discrimination tasks improves over the course of cognitive development (Feigenson et al., 2004). Furthermore, a well-known study by Landauer (1967) showed that when comparing the magnitude of two numbers, adults both took more time and produced more errors when the difference between two numbers were smaller. Therefore, instead of a precise bit-wise representation of numbers, it is believed that humans employ two systems: approximate representations that capture the relationship between numbers, where the accuracy is driven by their ration in magnitude, and an exact representation to capture small numbers (Feigenson et al., 2004). The exact representation can be extremely limited in early-staged of development – in second a study by Feigenson et al. (2004), they found that infants were able to reliably choose the larger of two pools of crackers where comparing amounts of 1 vs. 2 and 2 vs. 3, but chose randomly when given choices of 3 v. 4, 1 v. 4, 2 v. 4, and 3 v. 6, indicating a particular sensitivity to numbers lower than 3. This sensitivity to *value* of numbers, even when the number of digits is maintained, suggests a form of complexity unique to human behavior that drives the difficulty and performance on arithmetic operations as simple as subtraction and comparison.

While many basic findings regarding numerical processing, such as the distance effect and the numerical Stroop effect (confusion regarding the relationship of a digits magnitude and physical size when presented), have been replicated across different cultures, such as school children in the United States and Japan, there still exists strong cultural diversity across strategies for more complex arithmetic operations (Omura and Matsuta, 2018). One interesting example is *parity processing*, or determining in a number is odd or even, which has $O(1)$ time complexity when a number is represented as bits. Through a series of cross-cultural studies measuring auditory parity processing, Heubner et al. (2018) observe a general trend that parity processing is sensitive to properties such as shorter processing time if the number is square (perhaps due to square numbers being taught more in mathematics education), and longer processing time if the number is prime. However, they also observe linguistic sensitivity: in German, the digit relevant for parity judgement is pronounced first, leading to shorter reaction times than both English or Polish speakers. Furthermore, whether a number was part of the multiplication table of not led to different effects between German and Polish speakers, which was hypothesized to be due to different cultural attitudes towards using multiplication table drills in early mathematics education (Heubner et al., 2018).

Finally, mathematics education plays a strong role in the development of cognitive skills, and subsequent behavioral complexity, for operations such as multiplication or large summations. One famous example is the "Chinese lattice" method for multiplication, where a lattice grid is used to break down the steps of long-multiplication, and a similar approach called "Japanese Visual Multiplication". Teaching of these methods are often driven by the idea that spatial ability helps develop STEM expertise (Wai et al., 2009). In fact, there appears to be a strong human preference for spatial methods of solving mathematical problems, including an anecdote of the famous mathematician Carl Gauss to "fold" a list of numbers and sum the pairs when performing a summation over a list of consecutive integers (Hayes, 2006). Finally, Frank and Barner (2012) showed that humans trained to use an abacus to perform computations such as multiplication, division, and even square cube roots, as in many Asian cultures, develop a visual "mental abacus", appearing to manipulate imaginary beads in the air and solving problems with high speed and accuracy. Unlike a control set of participants, such users were insensitive to language interference when solving problems, suggesting that this visual representation helps reduce the overall task complexity of performing arithmetic operations (Frank and Barner, 2012).

7 Kolmogorov Complexity

Departing from time or space-based notions of complexity, another notion of computational complexity is Kolmogorov complexity, or the length of the shortest computer program that can produce a given output.

7.1 Overview

Kolmogorov complexity is naturally dependent on the programming language it is defined over, and was primarily studied by as a way of measuring the randomness of a sequence (). For example, one can consider a string s as truly random if every computer program that produces s is as long as s itself. There is also a natural connection with compression – specifically, if there exists a method that can compression a string s by c bits, then we can upper bound the Kolmogorov complexity of s with $|s| - c$.

7.2 Human behavior

One aspect of human behavior where Kolmogorov complexity provides a useful tool of analysis is understanding subjective perceptions of *randomness*: how do people judge the randomness of a sequence of events? This plays an important role in tasks such as memorizing phone numbers or Social Security numbers, as well as reasoning about sequential events (e.g. a tennis player's winning streak). One measure of human perception of randomness is the "subjective probability of alternation", or the subjective probability that an event in a sequence will have a different outcome than the preceding event, which has been shown to significantly differ from the actual event probability (Sun and Wang, 2012). In a well-known paper about randomness, Falk and Konold (1997) propose that this bias is due to the "difficulty level" of a sequence based on its minimum description, and therefore an underlying notion of Kolmogorov complexity. Through a series of three experiments, where human subject study participants were asked to study sequences until they could type them from memory, Falk and Konold (1997) found that the perceived notion of randomness was more correlated with this "difficulty level" measure (which measures the number of "runs", or subsequences that can be described by short expressions) rather than entropy, which would be the more standard, objective way of measuring randomness. However, Griffiths and Tenenbaum (2003) suggest that Kolmogorov complexity online provides too weak a set of constraints to properly capture human subjective judgements, and propose a statistical model to better capture priors such as symmetry and motif-replication.

Another domain where Kolmogorov complexity is relevant is analysis of human visual perception. For example, Schmidhuber (1997) introduced the notion of "low-complexity art", where an observer's preferred drawing from a larger set of drawings is the one with which the information required to computer their subjective model of aesthetics is minimized. Knitsch (2012) draws similarity between this notion and the idea of "gestalt", or perceiving objects as complex systems over small components, as well as the idea of visual "interestingness" stems from when there can be further compression – or complexity is higher. Finally, Peptenatu et al. (2022) recently performed an interesting study over 1200 paintings from of Byzantine art (from Greece, Russia, and Romania) that show that Kolmogorov complexity, implemented via the amount of memory required for a loss-less compression algorithm, helped differentiate the three different schools of art more so than entropy-based measures. These works suggest that the patterns inherent to visual perception – repeated motifs, blocks of color, shape structure – allow Kolmogorov complexity to be a useful explanation for human behavior both in appreciating and producing art.

8 Social Learning

While most of the case studies described above analyzed individual human behavior across different tasks, they did not focus on how humans *learn* such strategies, particularly via collaboration and social learning from others. Social learning enables the communication of optimal strategies between individuals, as well as leveraging the different abilities of a diverse team to discover a novel approach or algoirthm. A popular example of social learning is the Foldit game, where players collaborate to design protein structure models. Khatib et al. (2011) created a capability withing Foldit to create "recipes", or smaller algorithmic building blocks and strategies, which were then able to be shared, modified, and re-used by other players in the game. In fact, interviews by the research team showed that "word of mouth" was a common way successful recipes were shared, and eventually two new algorithms became dominant across all players ("Blue Fuse" and "Quake"). Meanwhile, scientists from the Rosetta project[2], developed new optimizations for their protein design algorithm, called "Fast Relax". When comparing "Fast Relax" and "Blue Fuse", Khatib et al. (2011) not only see comparable performance, but also in terms of the underlying building blocks, with both algorithms alternating the protein repulsive interaction strength from high to low repeatedly. These results show that even non-scientists, through communication and shared discovery, and help discover algorithmic strategies that perform close to state-of-the art.

Thompson et al. (2022) take a closer look at social learning by focusing on a more classical problem: sorting. They asked 3,450 human subject study participants (separated into 12 population groups) to sort unknown sequences of numbers, with a reward for performing fewer operations and incentives to look and upon prior participants' strategies. Just as in FoldIt, two algorithms emerged as dominant in frequency across all algorithms (80% of all participants): the first was selection sort, a well-known algorithm with time complexity $O(n^2)$, and the second was gnome sort, which is slightly more efficient (towards is $O(n)$) depending on the initial state). However, participants found it easier to describe, and therefore communicate their strategy, for selection sort, and therefore *language-based* cultural transmission resulted in a preference for selection sort, while participants learning gnome-sort primarily relied on visual descriptions (Thompson et al., 2022). These results suggest that human behavior is partially influenced by the ease in describing the strategy for a task. This is particular relevant when considering *teaching*, and ways we can develop instructional curricula to help teach optimal strategies. In a series of experiments involving a knot-tying task, Caldwell et al. (2017) show that the utility of having access to an expert teacher for instruction depends on task difficulty: while for simple knots, simply observing the target outcome led to equally high learning outcomes, for more complex knots (complexity measured by number of steps required), having a teacher describe

[2]https://www.rosettacommons.org/

the optimal approach was important. Overall, these results emphasize how human behavior in solving algorithmic tasks is not stand alone, but influenced by others.

9 Conclusion

Thus far we intentionally focus on simple algorithmic tasks and notions of complexity, the purpose is to better inform how different types of tasks result in if, and how, human behavior deviates from optimal strategies, as well as leads to heterogeneity amongst human subjects. However, more complex tasks such as Wordle or Chess (EXP time complexity) have been studied by complexity theoreists, and initiatives such as the MAIA chess project seek to replicate human-like strategies for playing chess show that sub-optimalities such as bounded human memory result in different kinds of behavior (McIlroy-Young et al., 2020). Likewise, Wu et al. (2018) show that Gaussian process function learning behavior explains human behavior in vast search problems where, under human's limited horizons, it is difficult to obtain the optimal strategy. As we seek to design technologies that better augment human capabilities, the goal of this overview is to provide better intuition for how computational strategies to solving classical tasks compare with those used by humans.

References

D. L. Applegate, R. E. Bixby, V. Chvatál, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006. ISBN 9780691129938. URL http://www.jstor.org/stable/j.ctt7s8xg.

C. A. Caldwell, E. Renner, and M. Atkinson. Human teaching and cumulative cultural evolution. 2017.

H. Chapman and M. Rupert. A group-theoretic appr etic approach to human solving strategies in sudoku. 2012.

G. Clarke and J. Wright. Scheduling of vehicle routing problem from a central depot to a number of delivery points. 1964.

G. Cooper. The computational complexity of probabilistic inference using bayesian belief networks. 1990.

R. Falk and C. Konold. Making sense of randomness: Implicit encoding as a basis for judgment. 1997.

L. Feigenson, S. Dehaene, and E. Spelke. Core systems of number. 2004.

M. Flood. The traveling-salesman problem. 1956.

P. Fontaine, F. Taube, and S. Minner. Human solution strategies for the vehicle routing problem: Experimental findings and a choice-based theory. 2020.

M. C. Frank and D. Barner. Representing exact number visually using mental abacus. 2012.

S. Gershman, E. Horvitz, and J. Tenenbaum. Computational rationality: A converging paradigm for intelligence in brains, minds, and machines. 2015.

T. Griffiths and J. Tenenbaum. Probability, algorithmic complexity, and subjective randomness. 2003.

B. Hayes. Gauss's day of reckoning. 2006.

L. Heubner, K. Cipora, M. Soltanlou, M.-L. Schlenker, K. Lipowska, S. M. Gobel, F. Domahs, M. Haman, and H.-C. Nuerk. A mental odd-even continuum account: Some numbers may be "more odd" than others and some numbers may be "more even" than others. 2018.

A. Karatsuba and Y. Ofman. Multiplication of many-digital numbers by automatic computers. 1962.

F. Khatib, S. Cooper, M. D. Tyka, K. Xu, I. Makedon, Z. Popović, D. Baker, and F. Players. Algorithm discovery by protein folding game players. 2011.

W. Knitsch. Musings about beauty. 2012.

D. Knuth. Dancing links. 2000.

P. Krolak, W. Felts, and G. Marble. A man-machine approach toward solving the traveling salesman problem. 1970.

R. S. M. . T. K. Landauer. Time required for judgements of numerical inequality. 1967.

N. L. Lee, G. P. Goodwin, and P. Johnson-Laird. The psychological puzzle of sudoku. 2009.

F. Lieder and T. L. Griffiths. *Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources.* Behavioral and Brain Sciences, 2020.

J. N. MacGregor and T. Omerod. Human performance on the traveling salesman problem. 1996.

R. McIlroy-Young, S. Sen, J. M. Kleinberg, and A. Anderson. Aligning superhuman AI and human behavior: Chess as a model system. *CoRR*, abs/2006.01855, 2020. URL https://arxiv.org/abs/2006.01855.

C. Murawski and P. Bossaerts. How humans solve complex problems: The case of the knapsack problem. 2016.

K. Omura and S. Matsuta. Numerical processing and executive functioning in early versus middle childhood: A japanese sample. 2018.

D. Peptenatu, I. Andronache, H. Ahammer, R. Taylor, I. Liritzis, M. Radulovic, B. Ciobanu, M. Burcea, M. Perc, T. D. Pham, B. M. Tomić, C. I. Cîrstea, A. N. Lemeni, A. K. Gruia, A. Grecu, M. Marin, and H. F. Jelinek. Kolmogorov compression complexity may differentiate different schools of orthodox iconography. 2022.

S. Sahni. Approximate algorithms for the 0/1 knapsack problem. 1975.

J. Schmidhuber. Low-complexity art. 1997.

A. Stuart. *The Logic of Sudoku*. Michael Mepham Publishing, 2007.

Y. Sun and H. Wang. Perception of randomness: Subjective probability of alternation. 2012.

B. Thompson, B. van Opheusden, T. Sumers, and T. L. Griffiths. Complex cognitive algorithms preserved by selective social learning in experimental populations. *Science*, 376(6588):95–98, 2022. doi: 10.1126/science.abn0915. URL https://www.science.org/doi/abs/10.1126/science.abn0915.

T. Torralva, E. Gleichgerrcht, A. Lischinsky, M. Roca, and F. Manes. "ecological" and highly demanding executive tasks detect real-life deficits in high-functioning adult adhd patients. 2013.

D. Vickers, T. Mayo, M. Heitmann, M. D. Lee, and P. Hughes. Intelligence and individual differences in performance on three types of visually presented optimisation problems. 2004.

J. Wai, D. Lubinski, and C. P. Benbow. Spatial ability for stem domains: Aligning over 50 years of cumulative psychological knowledge solidifies its importance. 2009.

C. M. Wu, M. S. Eric Schulz, J. D. Nelson, and B. Meder. Generalization guides human exploration in vast decision spaces. 2018.