2019

Drawing Parallels between Heuristics and Dynamic Programming

Margarita M. Dominguez

Drawing Parallels between Heuristics and Dynamic Programming

Margarita M. Dominguez

Bachelor's of Science in Computer Science, Psychology Minor

Seidenberg School of Computer Science and Information Systems

Advised by: Dr. Miguel Mosteiro

Presented: May 8th, 2019

Graduating: May 2019

Table of Contents

Abstract

Algorithms, or processes for doing something, are a topic that scholars and mathematicians explored long before computers were invented. As humans, we look to simplify our problems and take shortcuts. In psychology, this is called heuristic problem-solving, where humans use the outcomes of previous situations, to breakdown and decide how to approach new situations. This project explores the relationship between heuristic problem-solving and a computer algorithms, dynamic programming, in order to explore how humans rationalize these concepts and how we can extend the discussion of algorithms outside the technology field. Dynamic programming takes on the approach of breaking down a problem into smaller solutions, and then uses those smaller solutions as a reference for solving other computations and the overall problem. Those references created in dynamic programming are similar to people's personal memories. This project explores the relationship between heuristic problem-solving and computer algorithms, dynamic programming, in order to explore how humans rationalize these concepts and how we can extend the discussion of algorithms outside the technology field.

Introduction

Heuristics and algorithms are not topics typically thought of in conjunction. The idea of heuristics comes from the field of psychology and algorithms are discussed in computer science with their origination coming from mathematics. As a computer science major, I have always held a fascination for coding and the way a computer can translate the instructions it is given. Similarly, I hold a fascination for psychology and the way our brains work, which is how I ended up focusing on a way to compare those two topics.

Heuristics, like algorithms, aid in problem-solving and decision making. Those mental shortcuts guide us in our daily activities. Because of heuristics, we can usually figure out a solution, or at the very least decide how to proceed, when faced with a new situation. In our problem-solving, there are steps we take to assess the situation, and heuristics are a large part of this.

When an algorithm is solving a problem, making a decision, or simply working through a computation, it can be designed to take similar measures. Dynamic programming specifically is an algorithm that draws on a similar approach to heuristics. Dynamic programming algorithms are designed with the same intent to assess the overall situation, break that down into sub-problems, and find, and store, solutions to those sub-problems first. The storage of the sub-problem solutions is comparable to the way we store memories.

Looking at this research in a larger context, I think that if more people began to recognize how much of our own human thought process is in our computers, the field of technology, and the design and engineering behind it, could be accessible to people outside of the fields we typically regard as STEM. We can begin to discuss algorithms in a wider range of majors in our colleges and begin to bring a broader range of perspectives and knowledge into this topic.

Literature Review

Most of the sources I have been working with in my research have been journal publications. From the side of the psychology research, Ebsco databases were a big help. From the computer science side, the Institute of Electrical and Electronics Engineers (IEEE) database was the site of most of my findings. The topic of problem-solving on its own is one that has been explored in general. Human beings are constantly in situations where they are faced with making a choice - whether its at work, school, or down to a basic need such as choosing what they want to eat. For this reason, numerous psychologists have researched how it is that humans make decisions. The three largest contributors to this topic were Herbert Simon, Amos Tversky, and Daniel Kahneman. The three of them conducted a series of studies in this field - specifically in heuristics.

Starting with Simon's work, he, and his associates, wrote a paper titled "Decision Making and Problem-Solving" (1987). In this paper, Simon et al. explore problem-solving theory and decision making. This is important to the topic at hand because it offers an introduction and serves as a foundation on decision making and problem-solving. The study breaks down the importance of those two processes as separate entities. From the decision making side, Simon et al. discuss subjective expected utility (SEU), which is a sophisticated mathematical model of choice but could only work in a perfect world where humans were filled with more certainty (1987). The authors also found, through empirical research, that problem-solving is a process that is *heuristic* - meaning that it would be difficult to assess in a straightforward, scientific way. The empirical research was commonly conducted as asking people a series of questions, which was then supplemented with carefully designed laboratory experiments and observing how people make choices (Simon et. al, 1987). Similarly, problem-solving was studied in a controlled laboratory setting. This would then be supplemented with real-world examples such as a person playing chess. Simon et al. then dive into how these processes affect artificial intelligence (AI) research, how to deal with ill-structured/complex problems, the framing of problems, and then how computers handling computations is a form of problem-solving as well. Simon et al.'s work is very relevant and important to this analysis because he broke down the topic into its subproblems and analyzed how they would continue to play into technology in the future.

Additionally, Simon wrote another essay, in 1997, titled "Problem Solving". In this essay, he goes in depth about what it is, how problems are represented, and how problem-solving extends beyond the human mind. It is a field in cognitive psychology, but Simon wanted to explore other representations of problem-solving and how those heuristic rules apply outside of psychology. He also explored how heuristic analysis occurs and problems break down into subproblems. Both human and automated problem solving depends on the heuristic aspect and recognition of cues to access the knowledge necessary to solve the problem. This is an important point to note because the process of dynamic programming that I will be exploring also breaks larger problems into subproblems, stores the answers to those subproblems, and then uses them to as references to solve the overall larger problem.

Moving on to Amos Tversky and Daniel Kahneman's contributions, their focus was on decision making. The first paper of theirs that I will be using is "The Framing of Decisions and the Psychology of Choice" (1981). In this paper, Tversky and Kahneman address "the psychological principles that govern perception of decision problems" and how changing the way the problem is framed can significantly change the outcome (1981, p. 453). Tversky and Kahneman define what a decision problem is, how it is framed - meaning the conception of the decision maker's acts, the outcomes of that decision, and "contingencies associated with a particular choice" (1981). The focus in this paper is to explore human perception and overall thought process as it serves as a base for making decisions. They used hypothetical situations such as having subjects decide how many people to save from a deadly disease with a small percent chance that you could possibly save none of them at all in the process or how much money you can choose to gain if the choice is to gain a small amount or a big amount with the probability to actually gain nothing at all. This is a way to assess how we evaluate risk in decision making. The ultimate conclusions in their discussion highlight how perception is a large influence because of personal preferences, awareness of alternative solutions and frames, and confusion in how to resolve detected inconsistencies. This paper offers insight on the process of how humans rationalize decision making, which will be important in discussing making choices in general in problem solving and how this might tie into the rationalization of other concepts, such as dynamic programming.

One other work of Tversky and Kahneman is “Rational Choice and the Framing of Decisions”. This paper similarly analyzes how people frame and evaluate outcomes, by presenting the same types of decision making problems, but it goes deeper into the math of the outcomes of those choices. The purpose of it was to discuss how the descriptive theory for decision making could not be discussed solely on the foundation of the logic of choice because actual human behavior exhibits widespread deviations. One aspect of this publication that is very helpful is how Tversky and Kahneman breakdown the hierarchy of normative rules for decision making, which are four approaches humans take: cancellation, transitivity, dominance, and invariance (1981). Tversky and Kahneman argue that it is equally difficult to confine the process to this normative theory, but the hierarchy can still serve as a basis for human thought process even if the actual process does not directly confine itself to such categories. The two of them do a good job of bringing this cognitive, psychological process into a more technical light with their findings.

One last work I would like to mention from the heuristic side is an article from Science Direct, titled “Heuristics for Decision and Choice” (Todd, 2001). This is a short reading that serves as a source for defining heuristics in my writing being that I would like to provide a formal definition. “Heuristics for Decision and Choice” provides the reader with a quick summary of the way heuristics play into decision-making. It touches on the classical norms of rationality, how heuristics can exploit structured information, and how they play into judgment in situations where one must decide what outcome will occur based on the probability that it could occur.

Now, when it comes to the computer science side of this research, what literature exists for dynamic programming is not as straightforward as what has been written for the psychological processes. When it comes to dynamic programming, I chose two sources regarding its history and development, one written by the creator of dynamic programming, the other written by Stuart Dreyfus from Bellman's perspective. I am discussing only those two in order to provide a concise description of the subject and address the creator's intention and vision for it. The other sources I will mention are about specific applications of dynamic programming.

Starting with the "History and Development of Dynamic Programming", by Richard E. Bellman and E.S. Lee (1984). This paper outlines the history of dynamic programming and how it was developed. An interesting point noted in this essay is that the formulation of a scientific process, such as this, was not one that was straightforward; it was made to be "logical and rational" afterward, and this directly relates to our own nonlinear thought process when discussing cognitive problem solving. Dynamic programming was formulated as a solution to mathematical computations. Bellman formulated this process as an approach to multistage processes. The paper then goes on to discuss its application in deterministic problems and optimal control, numerical solutions in practical problems, invariant embedding, and the difficulty of dimensionality (as the variables in a problem increase). Bellman and Lee's essay finishes off with a short discussion and looking forward with dynamic programming and its projected future in artificial intelligence.

The next piece of literature is titled "Richard Bellman on the birth of Dynamic Programming" (Dreyfus, 2002). This essay was interesting because it was compiled by Stuart Dreyfus using Bellman's autobiography, *Eye of the Hurricane*, from 1984. Similar to the previous paper which discusses its origin, Dreyfus' piece goes through Bellman's introduction to multistage decision problems, the origin of the name "dynamic programming", optimal control theory, and his systematic methodological approach to mathematics. The whole essay is compiled of quotes directly from Bellman; therefore, it offers great insight into his thought process as he was creating the concept of dynamic programming and his vision for how it would play into problem solving beyond just computations. Dreyfus' compilation of Bellman's ideas brings a new perspective because he arranges Bellman's thoughts in a cohesive way and presents that narrative to the reader. It is interesting to see how another person is able to take Bellman's word and arrange them in a way that is still accurate.

Between those two pieces of writing, the reader can obtain a solid background on dynamic programming's beginnings from the man that created this idea. Additionally, those sources show us how an algorithm comes to be. I limit my discussion on background to those two sources in order to avoid repetitive information.

The next couple of readings serve as specific examples of applications of dynamic programming. Because dynamic programming is a technique for breaking down processes, it can be applied in endless scenarios. The method of memoizing, or storing, smaller solutions has proven to be a helpful solution. Specifically, within the world of programming, taking this approach to divide-and-conquer is more efficient than attempting to tackle through a problem with brute force.

The first is, "The Application of Dynamic Programming in the Nonlinear Programming Problems" by Qin et al. from 2010. This paper by Qin et al. which looks at the nonlinear programming dives into the math of these problems, but through the break down of the mathematical problems, one can compare that to the breakdown from a psychological perspective. Ni et al.'s work looks at how heuristics are actually used in dynamic programming. This paper is one that is far more technical than I believe I will use as it goes into using networks and mathematical computations, however, Ni et al. do address the structure of this type of algorithm, and looking at nonlinear problems will be helpful in forming a comparison to psychology because our own problem-solving and decision making does not always follow a direct path.

In relation to indirect paths, "Dynamic Programming for Fuzzy Systems with Fuzzy Environment" by J. F. Baldwin and B.W. Pilsworth in 1982. Although dated, this source serves as a great discussion piece when discussing how an algorithm can approach complicated, unclear situations, similar to the way humans do. Specifically, Baldwin and Pilsworth present us with "an equation for a multi-stage decision problem with fuzzy dynamics and environment" (1982, p. 1). They also mention human intelligence as part of their discussion - although they preface with writing that human intelligence is superior to the machines, a fact that was likely true at the time (our current machines and artificial intelligence are becoming smarter every day).

Additionally, Baldwin and Pilsworth note how human intelligence is limited in some ways, there are times when we over-simplify problems, which is why dynamic programming could be really useful as it could explore a wider range of solutions before making a decision (1982, p. 1-2).

Some other papers I would like to discuss, involve a heuristic approach to dynamic programming. This algorithm is simply called heuristic dynamic programming, or more often shortened to be HDP, and actually feature two of the same authors, which shows that the researchers working on this topic are working on it extensively thus the source becomes more credible. The first is "Reinforcement Learning Control Based on Multi-Goal Representation Using Hierarchical Heuristic Dynamic Programming" (Ne, He, et al., 2012). In this paper, Zhen Ni, Haibo He, Dongbin Zhao, and Danil V. Prokhorov look to create a hierarchy of heuristic dynamic programs as the way to break down a larger problem. Their intention is to use this hierarchical approach to store solutions, generate multiple goals, and allow the system to learn how to optimize itself in doing so. Next is, "Heuristic Dynamic Programming with Internal Goal Representation", written by Zhen Ni and Jaibo He (2013). In this paper, the two researchers analyzed another goal structure based on heuristic dynamic programming, but this time, they specifically wanted to address the "2-D maze navigation problem" (Ni and He, 2013, p. 2101). The latest of the ones I have chosen is "Model-Free Dual Heuristic Dynamic Programming" by Zhen Ni, Haibo He, Xiangnan Zhong, and Danil V. Prokhorov, written in 2015. Here the researchers ran a few studies using heuristic dynamic programming, which they describe as a "popular approach in approximating optimal solutions in control problems” (2015, p. 1834). The studies involved complex mathematical equations and solve for a variable, and they compare HDP to their proposed Dual Heuristic Dynamic Programming. Between these three sources, I am able to obtain substantial information on the design of this type of dynamic programming and what qualifies it to be considered “heuristic”. These different applications of heuristic dynamic programming present us with an idea of the range of problems this algorithm can address.

Overall, these are two topics that have been studied in detail separately. The challenges with this research come in with finding publications from each topic that can be compared. A majority of the written works regarding dynamic programming are extremely technical in terms of specific programming and mathematical computation concepts. The way dynamic programming came to be was as an idea developed by Richard Bellman and it was defined as it was implemented. Algorithms require formulas and numbers to be explained because it is important to mention complexity and how optimal the method is whereas a cognitive process can be observed and described in words. Numbers are important in psychology when it comes to discussing outcomes and providing a statistical analysis of the findings. In order to study an algorithm, however, researchers often, if not always, test them in mathematical scenarios.

Research Question

When learning about computer algorithms and programs, it’s interesting to think of them from a different perspective. Beyond the programming, there are other aspects that go into the design and implementation of algorithms. These algorithms follow specific steps and are designed to approach specific situations. Taking a psychological viewpoint to these algorithms, it is clear to see the human influence in these very technical topics. When humans are faced with making decisions, they often go through steps in outweighing the costs and mentally referencing the outcomes of previous similar situations. There are algorithms that similarly do this and store the results of problems as they are solved in order to apply those results to future iterations. The fact that similarities exist between cognitive problem-solving and algorithmic problem-solving is a given. Algorithms are created and designed by humans; therefore, it would be impossible for an algorithm to exist without having any semblance of our own thought processes and biases. My intention is now to draw parallels between heuristic problem-solving and decision-making, cognitive processes, and algorithms, with a specific look at dynamic programming.

Methodology

To explore the relationship between heuristics and algorithms, the main focus is to explore academic research papers. In conducting a meta-analysis of the two topics, it is important to see what professional works have been written about them. From the psychology side, the works of Simon, Tversky, and Kahneman are most important as they were the ones who made the most significant findings when it comes to decision making and problem-solving. For the computer science side, it is important to explore the beginnings and the design of dynamic programming, and how it all works to break down and solve problems. These are two processes that are not typically compared to one another, but there is potential to re-imagine the way we teach, discuss, and design technological processes such as dynamic programming, when clear links are made between the likeness they have with our own minds.

I will do an analysis of psychological works pertaining to human cognitive processes in relation to problem-solving - specifically heuristic problem-solving. Through the use of heuristics, which are mental shortcuts humans take to aid in daily life and problem-solving, humans are able to rationalize and approach new situations. Heuristics are when humans make associations to previous experiences in order to understand how to approach unknown situations and problems. We reference our memories in hopes of drawing comparisons. Between that and mentally breaking down problems, we are typically able to handle the smaller issues and work our way up.

The intention is then to analyze an algorithm called dynamic programming, which seeks to break down problems and store solutions of those subproblems in order to reference them later. In connecting psychology to this computational process, one can even gain insight into how humans rationalize such a process. It is important to remember that every aspect of technology has been designed by a human; therefore, it should be no surprise to find a strong relationship between our cognitive processes and a computer's technical ones.

Discussion

Heuristic Problem-Solving

As previously stated, heuristics help humans simplify daily life. To give a formal definition, from the *International Encyclopedia of the Social and Behavioral Sciences*:

> Heuristics are approximate strategies or 'rules of thumb' for decision making and problem solving that do not guarantee a correct solution but that typically yield a reasonable solution or bring one closer to hand. As such, they stand in contrast to algorithms that will produce a correct solution given complete and correct inputs. More specifically, heuristics are usually thought of as shortcuts that allow decisions (Todd, 2015).

Humans refer to heuristics throughout the day whether it is for basic day-to-day decisions, such as what to eat for breakfast, or more complicated task, such as problems we face at work. For example, we apply heuristics every day when we get dressed. If we look at the weather and see that it is raining, we know from previous experience that we should: grab an umbrella, wear a raincoat, put on rain boots, wear a hood, etc. We learned in the past that the rain will make us wet and uncomfortable; therefore, we plan ways to accommodate for it. Remembering this saves us time from trying to figure out how to avoid the rain, or avoid getting wet, for the most part, so that we immediately know how to handle it. Simple as that may sound, it is a heuristic.

These rules we establish for ourselves become integral to our existence. Without the ability to find ways to apply our memories to new situations, our lives would be exponentially much more difficult. Rather than making quick judgments, and breaking down problems, we would spend a longer amount of time struggling to understand the unknown, and likely spend more time in distress.

Looking first at problem-solving, a topic that many psychologists are constantly trying to understand, one can see how the mind aims to separate what we know versus what we do not know in order to reach some sort of resolution. “Solving a problem is transforming a given situation into a desired situation, or goal,” (1997, p. 1) as written by psychologist Herbert A. Simon, who was one of the top influential people when it comes to this topic. In Simon’s essay, titled, “Problem Solving”, he breaks down the steps for solving a problem. In order to problem solve, humans must generate a “representation”, or reference a previously generated representation (Simon, 1997, p. 1). These representations include a description of the situation, operators (actions) to change the situation, and “tests” to assess “whether the goal has been achieved” (Simon, 1997, p. 1). Using those three components we form the representation, and that representation serves as a guide to help us figure out how we should proceed in the situation. The use of a previously generated representation is an example of heuristics.

Although helpful, it is equally important to remember that heuristics do not guarantee solutions - real-life situations cannot be confined to linear paths and mental formulas. As Simon stated, “heuristics follow plausible paths that often find satisfactory solutions” (1997, p. 3). This is not to say that heuristics are completely unreliable, it is natural for us to apply heuristics where we can and to dissect situations in order to identify which parts of it we do know how to handle. The heuristics we store could be thought of as educated guesses, or running an experiment. Once we see the outcome of that previously unknown situation, our brains will store that as new information, and this becomes a new representation that can then be used in the future when we encounter a new unknown problem. Humans go through life cognitively addressing problem-solving in this way. There is an endless cycle of forming new representations in order to rationalize the world around us. We can then continue to form heuristics through the representations. That being said, problem-solving is not only a matter of rationalizing the unknown.

Another aspect that is important to discuss is decision making. Problems can go beyond finding a definite solution. It is possible that once we understand a problem, we then have to make choices about how to proceed - whether that is choosing how to lessen the severity of the problem or how to resolve the problem. This area of psychology in regards to making choices was largely studied by Amos Tversky and Daniel Kahneman. The two of them did multiple studies regarding the framing of decisions and psychology of choice.

According to Tversky and Kahneman, humans make decisions based on the “psychological principles that govern the perception of decision problems and the evaluation of options” (Tversky and Kahneman, 1981, p. 453). This means our personal preferences and biases have a large influence over the way we perceive a situation. Our perception, and personal heuristics, ultimately explain why we make the choices we do. As written by Tversky and Kahneman, “The frame that a decision-maker adopts is controlled partly by the formulation of the problem and partly by the norms, habits, and personal characteristics of the decision-maker” (1981, p. 453) - this, we will later see, directly correlates with the biases that exist in our algorithms. It is very difficult, if not impossible, for humans to eliminate their personal biases. This means that the same problem presented to different people could result in completely different outcomes from one another because it is going to be perceived from each individual's unique perspective.

In saying that our interpretations influence the choices we make, that does not mean that the framing of choices is a completely random process. Humans assess both the costs and possible outcomes in order to decide if a situation will result in an outcome they find favorable, given the initial effort they must contribute. The outcome is defined through the expected utility model, and people usually look to obtain the "highest expected utility" in decision making (Tversky and Kahneman, 1981, p. 453). The expected utility model is "based on a set of axioms" (Tversky and Kahneman, 1981, p. 453). Those axioms are unique to each individual - we have our own preferences and value outcomes based on our own likes and personal situation. We then use those axioms to compare the utility of the outcome and the probability of that outcome occurring. Research has found, however, that even so, humans do stray from this expected utility model. The descriptive theory proposed by Kahneman and Tversky divides the decision-making process into two phases: the initial phase where we frame the outcomes and then the evaluation phase (1981, p. 453). This idea of two phases will become relevant again later when discussing heuristic dynamic programming, which similarly consists of two layers.

Going off the concept of expected utility theory, there are four principles of choice that are said to underlie this theory: cancellation, transitivity, dominance, and invariance. Cancellation is the action of eliminating occurrences that yield the same outcome rather than continuing to ruminate and worry about making a choice (Tversky and Kahneman, 1986, p. 252). Transitivity refers to the idea our preferences can change depending on the situation and which choice will yield a greater outcome at that given time (Tversky and Kahneman, 1986, p. 253). Dominance is simple, this is the idea that people will choose the option that is better given the current state, and is typically better in all other states. The last principle, invariance refers to the idea that regardless of the description, or "representation" of the problem, we should have the same preferences, therefore, make the same choices no matter how the problem is presented (Tversky and Kahneman, 1986, p. 253). I would argue that these four principles do hold importance in our decision-making, however, perceptions and heuristics do play a role. It is far too definitive to say that we will hold the same preference because as humans we are constantly evolving, learning, and forming new memories.

All of that being said, making choices plays into problem solvings. Even if a problem requires some sort of computation, people can be faced with making a decision. One could even say that problem-solving is an important step in making a choice because we mentally apply the expected utility model to calculate which choice will reap the greatest benefit. Where heuristics come in is in guiding us to avoid making redundant calculations if we already know what the outcome will be from previous experience, or help us eliminate a choice we already know we will not like the outcome to. These shortcuts help us save time, and we actually apply heuristics constantly - they are rules that are second-nature to us.

Algorithmic Problem-Solving

Similar to heuristics, algorithms are rules. According to Cormen et al, authors of *Introduction to Algorithms*, algorithms can be defined as:

> any well-defined computational procedure that takes some value, or set of values, as input and produces some value, or set of values, as output. An algorithm is thus a sequence of computational steps that transform the input into the output (2009, p. 5)

Algorithms can be applied in a wide variety of situations, whether computational or mathematical. They define a clear procedure to follow in order to complete the task, or problem, at hand. Algorithms can be thought of as tools that simplify the problem-solving process and help us work through computations. They help us save time, and often allow us to work more efficiently. In an algorithm, the input is explicitly defined, as is the output, and algorithms are effective for this reason. With an algorithm there is no confusion or ambiguity; in whatever algorithm you, or your computer, are working with the steps are clearly defined. Depending on the algorithm, it is possible to have to go through multiple iterations, repeat steps, until the outcome is obtained.

Algorithms can vary in their structure as well. An algorithm could look like a list of steps, similar to a cooking recipe, or it can go on to be as complex as implementing multiple loops, checking for satisfied conditions, solving multiple mathematical equations breaking down larger problems, storing values, and going through multiple iterations of the same steps.

That being said, algorithms should not only be thought of as big math problems. They are embedded in all of our technology to run a multitude of tasks. Algorithms come into play when a user is searching for an app, and they come into play when a user is asking a voice-assist program, such as Siri, a question. There are many specialized algorithms, but I will be focusing on one: dynamic programming.

Dynamic programming is an algorithm that can be applied to many different problems. This algorithm implements the approach to divide-and-conquer tasks. The problem the algorithm is trying to solve is divided into subproblems; the solutions to those subproblems are then used to compute the overall greater problem. There are four basic steps to follow in designing a dynamic programming algorithm:

1. Characterize the structure of an optimal solution.
2. Recursively define the value of an optimal solution.
3. Compute the value of an optimal solution, typically in a bottom-up fashion.
4. Construct an optimal solution from computed information.

(Cormen et al, 2009, p. 359).

Step 1 entails analyzing the problem and establishing what the best solution would consist of. In step 2, that solution is described recursively. Recursion is the process of defining something using itself in the definition. This means that a recursive algorithm iterates and repeats a part of itself until it finds the solution. Because a recursive process repeats itself, it typically works in a bottom-up fashion. Step 4 is not always executed. Recursion is key in dynamic programming because the program is solving the overall larger problem by doing multiple computations of subproblems. Each time a new solution is found, that value is then stored in a data structure - typically an array or hash table. The process of using a data structure to store data is *memoization*. Some problems are just computations to produce some value, and some other problems need to take that value to make a cohesive solution.

Interestingly enough, dynamic programming was created as a mathematical process by Richard Bellman, "gradually formulated" between 1948 and 1952 (Bellman and Lee, 1984, p. 2). Dynamic programming was intended for solving random multistage decision problem - random meaning random probability distribution (Bellman and Lee, 1984, p. 2). The name originated from wanting to mask the fact that it involved mathematics. The word dynamic was used because it is a dynamic process to go through the iterations, and it would grab people's attention. To find that dynamic programming came from math is not entirely surprising, however, because math has had a large impact on computer science, being that our data is processed as numbers.

Some specific examples of the applications of dynamic programming include: searching through a binary tree, cutting a rod into pieces for the lowest total cost, and producing values for a table. Dynamic programming is very helpful in multi-stage processes; at each step, a decision is made in order to optimize the result (Qin et al., 2010, p. 236). The optimized results to those subproblems are used to avoid redundant computation. Here you can see how dynamic programming and heuristics begin to tie together.

A heuristic form of dynamic programming has already been developed. Heuristic dynamic programming similarly tackles on an endless range of problems.

Drawing Parallels

Both in heuristics and dynamic programming, the focus is on identifying the problem, dividing the problem into subproblems, finding a solution to those subproblems, and using those solutions to solve the overall. Additionally, both processes allow the use of previously stored (sometimes calculated) information, or memories.

Cognitive Limitations vs Programming Limitations

Until now, the implications for both heuristics and dynamic programming is that they can be applied to a multitude of problem, regardless of the structure the problem seems to have. Both of these processes try to break up the problem to gain a better understanding. The human brain may not so explicitly define the steps it is taking upon encountering a new situation, but it does make some quick choices in order to choose well and choose efficiently. One big limitation humans encounter that algorithms do not is selective memory. Humans can be very forgetful, and as we get older, our memories can begin to blur. It would be impossible to remember the outcome of every single situation. This is where the algorithms can be regarded as superior. While storage and computational capacity are limited, what information is stored is not very likely to become blurry and unclear, but it is important to remember that our intelligence, in general, is limited; therefore, the algorithms we design will be limited as well.

Memories vs Memoization

Our memories are comprised of the experiences we have had throughout our lives. These memories range from precious moments with our loved ones to the subway routes one can take to get to work to the multiplication table we memorized in the third grade. Everything is information we can remember and use in the future when we are uncertain about how to solve a problem and what choices to make. For example, if a person on the subway misses their stop and they do not know how to walk to the destination from that next stop, they know to revert back to the stop they do know, and proceed from there. Dynamic programming implements a very similar approach. A program does not have memories, however, it has information that is stored. In the case of the subway example, the program would have the subway stops it has already visited stored in the order in which they occur. From there, the program would be able to see the link between the stops and direct the user to take a train in the opposite direction to arrive at the desired stop. In this case, the human brain can be considered to be equivalent to a hash table.

Heuristic Dynamic Programming

During this research, it was exciting to find that others have found a connection between heuristics and dynamic programming as well. Even more exciting was finding programs that implemented these two concepts together. What characterizes heuristic dynamic programming is the use of neural networks, specifically, two of them: an action network and a critic network (Tang and Srikant, p. 1). The use of the two networks mimics the human process. The critic network analyzes the situation and “learns the relationship between a set of control signals and the corresponding strategic utility function” (Tang and Srikant, p. 1). The action network uses the feedback from the critic network to produce the best solution.

Similar to when humans problem-solve, this type of dynamic programming will assess the situation, compare what they do know to what they don't, and assess the expected utility when faced with making a choice. Heuristic dynamic programs look at the overall costs in order to work efficiently, and these programs can be adapted to any type of multi-stage process. It is especially useful tasks that require the program to learn. Through the splitting of tasks and storing data, heuristic dynamic programs could help greatly with machine learning and be applied to artificial intelligence.

Discussing Algorithms

Heuristics and dynamic programming exhibit direct correlations to one another. It should be expected that the technology around us is going to function in a similar way humans do - especially as it increases in complexity and is created to execute more and more tasks for us. The devices we use every day are incredibly smart. Our smartphones alone act as organizational planners for our entire lives. There exists an app to fill almost any need we might have and most of us would likely have - even to do a task as simple as reminding us to drink water. Technology is becoming deeply embedded in our lives, beyond casual everyday uses. Computers and computer algorithms are being used in almost every job field. If not used directly, algorithms, at the very least, run the software behind the technology.

Because an algorithm such as dynamic programming mimics our own likeness so well, we are able to rationalize it. It is important to acknowledge this fact and realize that algorithms do not have to be discussed in the typically technical way they always are. Algorithms are instructions or rules to find a solution. Yes, when it comes to implementation, there will be math to consider because of run-time and the architecture of the device executing the algorithm. If these algorithms are created for humans to use them, they should be discussed outside typical STEM fields. The importance of interdisciplinary teams should not be disregarded. Imagine an anthropologist working with an engineering team for smart cars, or an economist developing an algorithm for trading money and stocks. Algorithm design and analysis should be taught to a wider range of students in order to open up the capabilities of what we can continue to do with technology. Humans are limited by their own cognitive abilities as it is, but if a wider range of people is given the knowledge, that increases the range of what we can accomplish. Just from conducting this research, the vast majority, if not all, of the paper's regarding algorithms were very technical and dove into the complex math behind these problems. It was impossible to read about how the algorithms in discussion worked, without having to decipher the equations. This shows that there is an unaddressed need for algorithms to be discussed in a new way.

Conclusion

In the end, both algorithms and heuristics are complex topics, and each one has a likeness to the other. Heuristics are the mental shortcuts and rules of thumb we follow on the day to day basis. They help us save time throughout our daily routines, and they help us when we find ourselves faced with a problem or decision to make. Through heuristics, we break down problems and reference our memories of previously encountered events in order to see the optimal way to approach. In dynamic programming, the algorithm proceeds in a similar way. As it executes, the program stores the information to the smaller problems it has computed in order to use that as a reference going forward. Those smaller references, comparable to memories, save the algorithm time from having to compute repetitive information and serve as a guide to complete the overall task at hand.

The clear similarities between these two topics help show us that algorithms apply similar methods as we do because they are human created. It would be impossible for humans to completely eliminate bias. When the algorithms are created, what is occurring is that the creators' thought-process is now translated into a format that is compatible with technology and coding. That being said, the discussion and teaching of algorithms should be taught to a wider range of people outside the STEM field, and even within the STEM field, algorithms should be more commonly discussed in a way that is not always technical and lending itself to be occupied with math and runtime complexities. At the end of the day, algorithms are simply instructions to follow. Mentally, we give ourselves instructions to follow every day. There could be a great deal of knowledge to untap if algorithms were introduced to those not involved in STEM.

References

Baldwin, J. F., & Pilsworth, B. W. (1982). Dynamic programming for fuzzy systems with fuzzy environment. *Journal of mathematical analysis and applications*, *85*(1), 1-23.

Bellman, R., & Lee, E. (984). History and development of dynamic programming. *IEEE Control Systems Magazine*, *4(4), 24-28*.

Cormen, T. H., Leiserson, C. E., & Rivest, R. L. (2009). *Introduction to algorithms*. MIT Press. Retrieved from http://search.ebscohost.com.rlib.pace.edu/login.aspx?direct=true&db=cat01787a&AN=PUC.b1411869&site=eds-live&scope=site

Dreyfus, S., (2002) Richard Bellman on the Birth of Dynamic Programming. Operations Research 50(1):48-51. https:// doi.org/10.1287/opre.50.1.48.17791

Ni, Z., & He, H. (2013). Heuristic dynamic programming with internal goal representation. *Soft computing*, *17*(11), 2101-2108.

Ni, Z., He, H., Zhao, D., & Prokhorov, D. V. (2012, June). Reinforcement learning control based on multi-goal representation using hierarchical heuristic dynamic programming. In *The 2012 International Joint Conference on Neural Networks (IJCNN)* (pp. 1-8). IEEE.

Ni, Z., He, H., Zhong, X., D. V. Prokhorov, D.V., "Model-Free Dual Heuristic Dynamic Programming," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 8, pp. 1834-1839, Aug. 2015. doi: 10.1109/TNNLS.2015.2424971

Qin, T., Huang, W., and Wang, G., "The Application of Dynamic Programming in the

Nonlinear Programming Problems," *2010 WASE International Conference on Information Engineering*, Beidaihe, Hebei, 2010, pp. 236-239. doi: 10.1109/ICIE.2010.151

Simon, H. (1955). A Behavioral Model of Rational Choice. *The Quarterly Journal of Economics, 69*(1), 99-118. Retrieved from http://www.jstor.org/stable/1884852

Simon, H. A. (n.d.). *Problem Solving* [PDF]. Carnegie Mellon University Digital Collections (pp. 1-9).

Tang, K. W., & Srikant, G. Reinforcement Control via Heuristic Dynamic Programming.

Todd, P. M. (2001). Heuristics for Decision and Choice. In *International Encyclopedia of the Social & Behavioral Sciences* (Vol. 11, pp. 6676-6679). Amsterdam: Elsevier. Retrieved from https://www.sciencedirect.com/science/article/pii/B008043076700629X?via=ihub.

Tversky, A., & Kahneman, D. (1986). Rational choice and the framing of decisions. *The Journal of Business, 59*(4), S251-S278. Retrieved from http://www.jstor.org/stable/2352759

Tversky, A., & Kahneman, D. (1981). The framing of decisions and the psychology of choice. *Science*, *211*(4481), 453-458.