1. Introduction and Literature Review: So what the heck is a genetic algorithm, anyways?

A genetic algorithm is a process that uses the principles of evolution and natural selection to develop solutions to computational problems. Genetic algorithms, along with neural networks, swarm-based algorithms, and other biologically-inspired algorithms, have been found in recent years to be surprisingly adept at devising efficient solutions to computationally difficult problems with little human intervention. However, because the notion of a "genetic algorithm" is a very technical concept and is by no means common knowledge, the literature review is preceded by an overview of the theory and basic mechanisms behind genetic algorithms.

1.1. Evolutionary Underpinnings: Why we have fuzzy bunnies

Imagine a litter of baby rabbits living in a frigid tundra. During their development, they received DNA from each of their parents, but the DNA wasn't copied perfectly. Some random errors might have occurred, causing each child to be slightly different. If a few rabbits underwent a gene mutation causing them to be, say, slightly hairier than their siblings, those few will be better suited to survive in their chilly environment, making them more likely to survive, find mates of their own, and pass down their mutated genes to their babies. Their grandchildren will inherit this "hairiness gene" and be abnormally hairy like their parents, giving them an advantage over their less-hairy peers, and so on. It is statistically likely that after hundreds of generations, the hairiness gene will become predominant, filling the tundra with fuzzy bunnies.

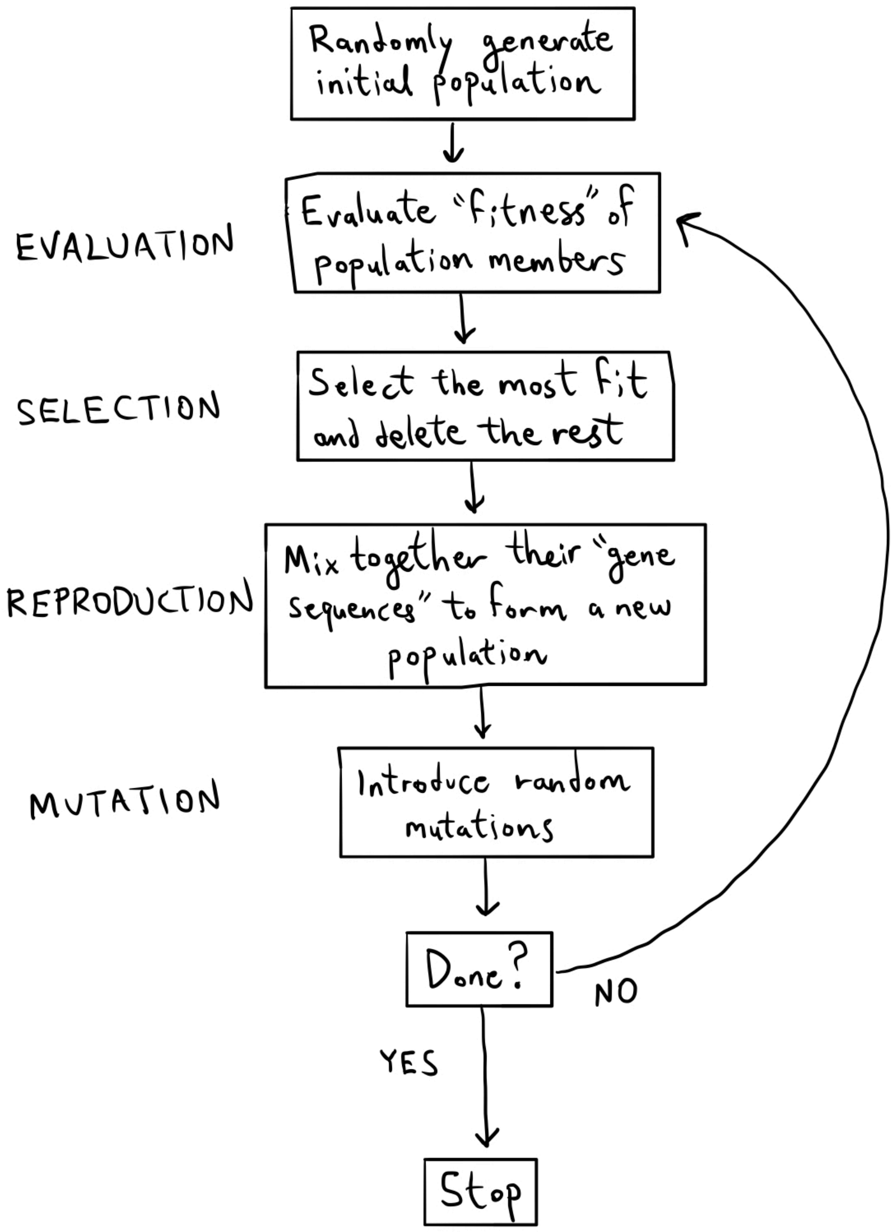
In *The Origin of Species*, Charles Darwin (1859) marvels over this process, which he calls "natural selection." Because of our more sophisticated modern understanding of genetics, biologists nowadays describe this process more precisely than Darwin did. Grant (1991), in an article published in *Scientific American*, defines natural selection as "differential success," or the process by which small, random adaptations increase the fitness of some organisms (p. 82). Gould, Keeton, and Gould (1996), two of whom were professors at Princeton University and Cornell University, define an organism's fitness as its "probable genetic contribution to succeeding generations," and define an adaptation as "a favorable characteristic that increases an organism's chances of perpetuating its genes, usually by leaving descendants" (p. 373). In nature, adaptations abound: Darwin (1859) lists diverse examples, like bird plumage used for courtship (p. 116), nectar used by plants to attract pollinators (p. 118), and the astoundingly complex human eye (p. 144).

Through natural selection, biological mechanisms have developed that are far more complex than any manmade machine, yet "plainly bear the stamp of far higher workmanship" (Darwin, 1859, p.113). Humans benefit from natural selection not only through adaptations of our own (like our massive brains), but also by breeding domesticated animals and selecting for favorable characteristics like strength and speed in horses (Darwin, 1859, p. 124). However, as Darwin (1859) puts it, "natural selection will always act with extreme slowness" (p. 123) since it takes place over the course of generations, preventing us from fully harnessing its power of intricate design. That is, until the invention of the genetic algorithm.

1.2. The Genetic Algorithm (GA): Artificial animal husbandry

John Holland (1992), a pioneering researcher who helped develop genetic algorithms, describes how they work in an article for *Scientific American*. Given some difficult puzzle or task, different strategies can be encoded in strings of numbers (like a gene sequence) forming what is called the "solution space," or the set of all possible encoded solutions to the problem. A virtual "population" of solutions (most of which will be inept at first) are generated, then tested to see which, if only by chance, perform best at the task in question. The worst solutions are eliminated, while the strongest ones are preserved, mixed together in certain ways (emulating "reproduction"), and tweaked with random mutations, adding the element of chance that drives evolution. These children become the new population, and the process is repeated, often thousands of times. The result is, as Holland (1992) puts it, "programs that solve problems even when no person can fully understand their structure" (p. 66).

The general structure of a genetic algorithm has become very standardized, and many researchers presenting applications of genetic algorithms describe their structure using flow charts very similar to the following (see, for instance, Parvez & Dhar, Kim & de Weck, Kumar & Paneerselvam):


Randomly generate initial population
EVALUATION
Evaluate "fitness" of population members
SELECTION
Select the most fit and delete the rest
REPRODUCTION
Mix together their "gene sequences" to form a new population
MUTATION
Introduce random mutations
Done?
NO
YES
Stop

Melanie Mitchell, formerly a researcher at MIT and the author of the book *Complexity: A Guided Tour*, defines a few relevant terms in her book *An Introduction to Genetic Algorithms*. In a genetic algorithm, the "population" is defined at the set of possible solutions being tested, each of which is determined by a sequence of numbers called "genome." "Evaluation" is the phase of the algorithm during which each member of the population is tested to determine its "fitness," which is defined as the value of some numerical function depending on the problem being solved (in this paper, any mention of the "quality" of a solution refers to its fitness). "Selection" is the process of choosing the fittest members of the population, “reproduction” or “recombination” is the process of combining the gene sequences of those selected to form a new population (which can be done in a number of different ways, as shall be discussed later), and “mutation” is the process of introducing random changes into the new gene sequences (Mitchell, 1996, p. 5-8).

Because this takes place on a computer, thousands of generations can go by in the blink of an eye, producing “clever” solutions to difficult problems without an ounce of human ingenuity required.

1.3. Applications: What do bunnies and jet engines have in common?

Genetic algorithms have been applied to various difficult mathematical problems and even produced solutions surpassing those designed by humans. Al-Sultan, Hussain and Nizami (1996) use a GA to quickly obtain approximate solutions to the “set-covering problem”, which is computationally difficult and time-consuming to solve in exact form. Mitchell, Crutchfield, and Das (1996) show how GAs develop a counterintuitive method of globally sharing local information in cellular automata. Simpson (1997) finds that a GA’s solution to the “bridge club scheduling problem” outperforms the four other dominant solution algorithms to which he compares it, in terms of solution quality and efficiency. These findings together confirm that the mindless genetic algorithm can often display more mathematical “ingenuity” than humans.

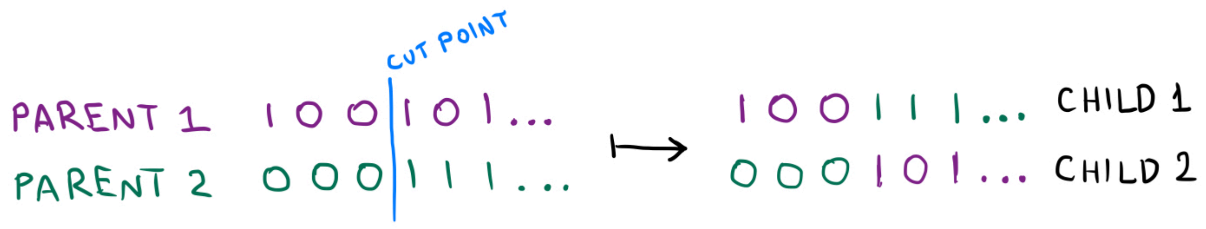
The genetic algorithm is not simply an esoteric artifact whose power is limited to solving math puzzles, and it has been applied to diverse real-world problems. Metawa, Hassan, and Elhosney (2017) show how GAs can be used to efficiently automate bank lending decisions by accepting data about the loan applicant as input and generating an optimal loan decision. Xiao (n.d.) designs a GA to solve spatial partitioning problems and postulates an application to political redistricting. Genetic algorithms have even proven useful when it comes to designing new technologies. Holland (1992) explains how GAs were combined with physics simulators to design optimal jet engine turbines. Various researchers have applied GAs to develop movement and path planning in mobile robots, like Achour and Chaalal (1996) and, more recently, Bezák (2012) and Sedreh and Zadeh (2018). Genetic algorithms’ plentiful applications need not even be restricted to the private sector: Beligiannis, Moschopoulos and Likothanassis (2009) applied GAs to the scheduling of Greek schools and found that the method they developed was a great improvement on the standard method due to its customizability.

Genetic algorithms are clearly a powerful tool driving mathematical discovery, private profit, public benefit, and technological innovation. In the next section, we shall see how changing certain parameters of a GA can alter its efficiency or the quality of its final solution. Because GAs have such wide-ranging applications, any improvement made to the structure of a genetic algorithm has the potential to enhance any of its implementations listed above.

1.4. Area of Inquiry and Research Question: When two arrays love each other very much...

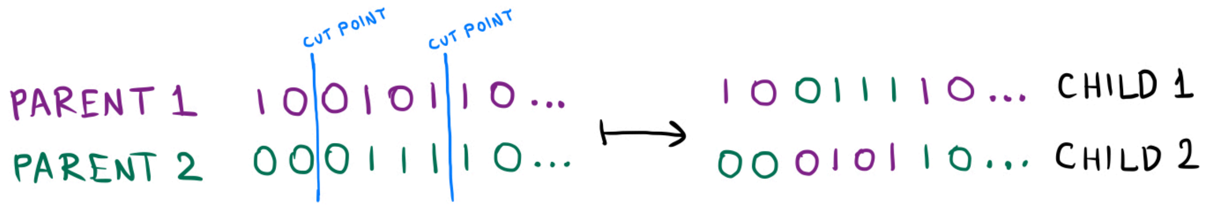
In an earlier section, “reproduction” was defined as the process through which the selected genomes of a population are combined to form a new population. However, there are many different ways of “combining” strings of digits. Two parents could be chosen from among the selected and their genomes spliced at a randomly chosen point, as shown in Figure 1:

FIGURE 1



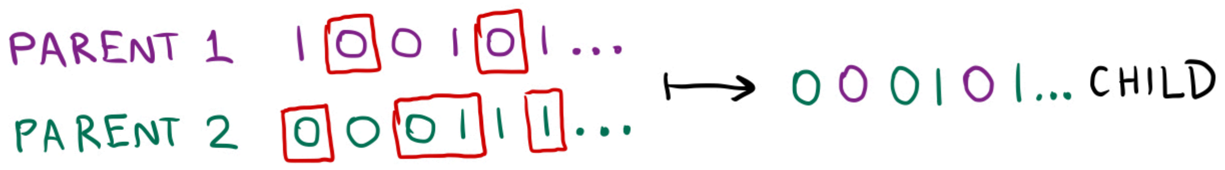
...or their genomes could be spliced using two randomly chosen cut points, as in Figure 2:

FIGURE 2



...or a child genome could even be formed by randomly choosing a character from either of its parents at each position in the sequences, as seen in Figure 3:

FIGURE 3



Each of these different methods is called a “crossover operator” (abbreviated CO). Umbarkar and Sheth (2015) and Kora and Yadlapalli (2017), in identically titled papers, list various crossover operators. It happens that the crossover operator showcased in Figure 1 is called “single-point crossover,” the CO in Figure 2 is “two-point crossover,” and that in Figure 3 is “uniform crossover.” Their papers include many different and more elaborate crossover operators, some of which even involve more than two parents.

The effect of the CO used on the the efficiency of a GA and the quality of its solutions is not fully understood. Several researchers have applied multiple different COs to the same problem and compared the results, but there exists no clear consensus. Picek and Golub (2010) compared the performance of many COs and remarked (despite not finding any conclusive results) that the GAs with uniform or two-point crossover typically performed best, while those with single-point crossover or no crossover at all typically performed worst. However, Magalhaes-Mendes (2013) compared four different COs and found that uniform crossover performed second-best, supporting the results of Picek and Golub (2010), but they ranked single-point crossover highest, contradicting Picek and Golub (2010). These results are not necessarily incompatible, but rather suggest that there is no unique “best” CO, and that different COs may be better suited for different problems depending on the topology of their “solution spaces.” Pujlic and Manger (2013) confirmed this by applying each of eight different COs that performed well on the “travelling salesman problem” (TSP) to the “vehicle routing problem” (VRP), discovering that “the obtained relative ranking of operators is quite different” (p.374).

This does not mean that the CO should be ignored, as various researchers have shown that is can have a considerable effect on the GA’s outcome. Both Emmanouilidis and Hunter (2000) and Ortiz-Boyer, Hervás-Martinez, and García-Padrajas (2005) specifically designed COs for problems involving neural networks, with promising results: the former noted that their GA found more viable solutions than the typical GA with n-point crossover, and the latter showed that their GA actually outperformed the traditional GA. Eiben and Raué (1994) also insightfully noticed that COs that recombined the genomes of three or more parents for each child often outperformed GAs with standard two-parent crossover.

This suggests the following question: how does the crossover operator used in a genetic algorithm affect the quality of the solutions obtained? Of course, it is impossible to answer this question in general, since the optimal CO depends on the problem to which the GA is applied, as explained above. However, the purpose of this paper is to shed some light on the issue by applying GAs with a variety of different COs to a two-dimensional spatial navegation problem, and analyzing the results using both statistical methods and evolutionary reasoning.