Abstract: A genetic algorithm (GA) is a process that uses the principle of evolution and natural selection to computational problems. GAs have been found to be surprisingly adept at devising efficient solutions to computationally difficult problems with little human intervention. In this paper, we begin by explaining the basic mechanism behind GAs and survey some applications. Later, we take a closer look at the reproduction step of the GA and explore the effect that a GA's crossover operator (CO) has on the performance and diversity of its population by applying GAs with a variety of COs to a simple puzzle. We conclude by noting the relationship between performance and diveristy (while recognizing that results obtained here may not be completely generalizable) and developing a heuristic combinatorial method of evaluating a CO's "creativity."

1. Introduction and Literature Review

1.1. Evolutionary Underpinnings: Why we have fuzzy bunnies

Imagine a litter of baby rabbits living in a tundra. During fertilization, they received imperfectly copied DNA from each parent. Small random errors might have occurred, making each child slightly different. If some underwent a mutation making them, say, slightly hairier than their siblings, those few will be better suited to their chilly environment and therefore more likely to survive, find mates, and pass down their mutated genes to their offspring. Their grandchildren will inherit this "hairiness gene" and also gain 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.

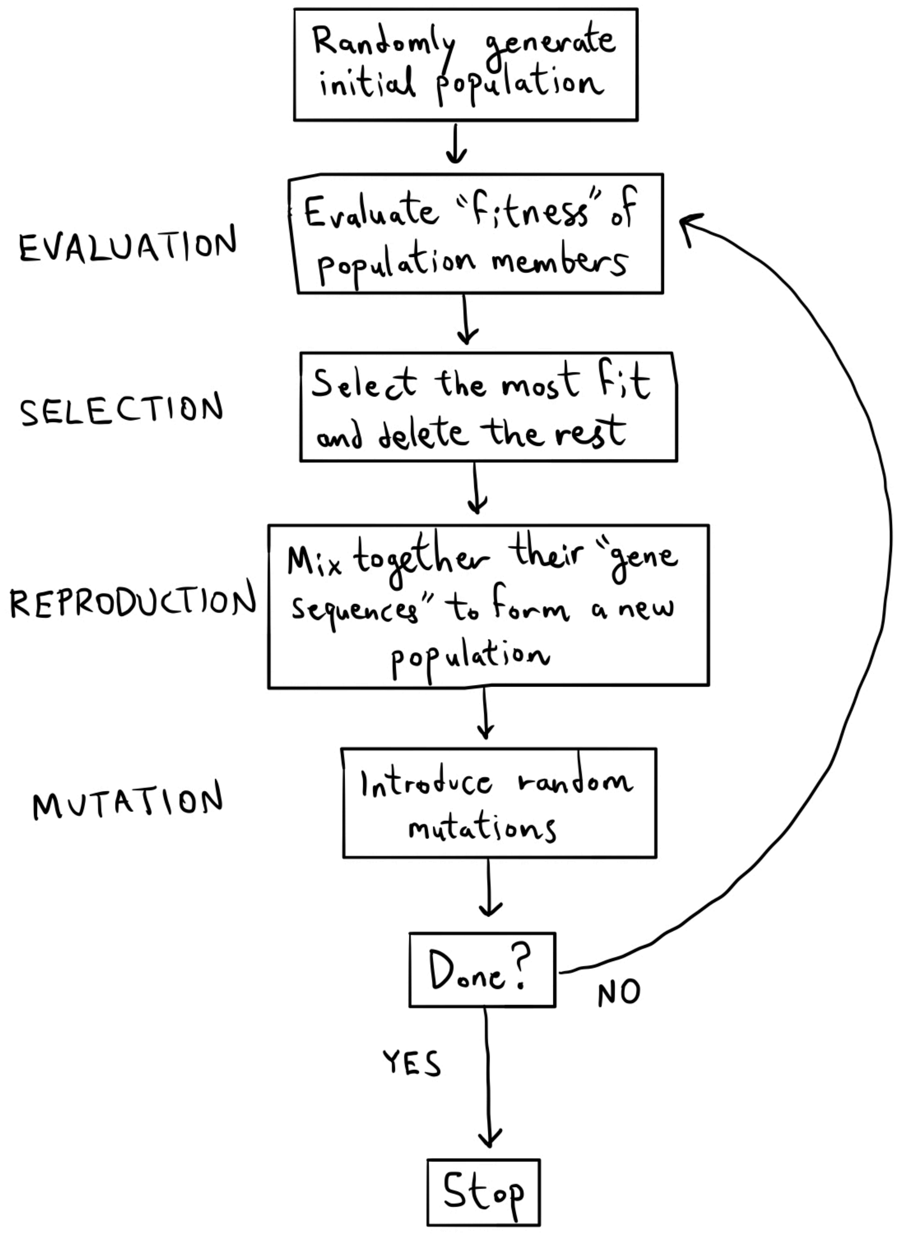
In *The Origin of Species*, Charles Darwin (1859) marvels over this process, which he calls "natural selection." Because of our more sophisticated modern understanding of genetics, biologists nowadays describe this process more precisely than Darwin. Grant (1991), in an article for *Scientific American*, defines natural selection as "differential success," 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). Adaptations abound in nature: Darwin (1859) mentions bird plumage used for courtship (p. 116), nectar used by plants to attract pollinators (p. 118), and the astoundingly complex human eye (p. 144).

Evolution has produced biological mechanisms far more complex than any manmade machine, seemingly "[bearing] the stamp of far higher workmanship" (Darwin, 1859, p.113). Humans benefit from evolution not only through our own adaptations, but also by breeding domesticated animals and selecting for favorable characteristics like strength and speed in horses (Darwin, 1859, p. 124). Unfortunately, as Darwin (1859) puts it, "natural selection will always act with extreme slowness" (p. 123) since it occurs over the course of generations. By speeding up this process in a virtual environment, genetic algorithms help us harness the power of evolution's designerless design.

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

John Holland (1992), a pioneering GA researcher, describes how they work in a *Scientific American* article. Different strategies for solving a difficult task can be encoded in strings of numbers (like gene sequences), together forming the "solution space," or the set of all possible encoded solutions to the problem. A virtual "population" of solutions is generated, then each is tested to see which perform best at the task in question, if only by chance. The worst solutions are eliminated, while the strongest ones are preserved, intercombined (simulating reproduction), and tweaked with random mutations, adding the random variation required for "differential success." These children become the new population and the process is repeated, often thousands of times. This results in what Holland (1992) calls "programs that solve problems even when no person can fully understand their structure" (p. 66). Because this takes place on a computer, thousands of generations can go by in the blink of an eye, quickly producing "clever" solutions without an ounce of human ingenuity required.

The general structure of a genetic algorithm is standardized, and many researchers applying genetic algorithms describe their structure using flow charts 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 MIT researcher and the author of *Complexity: A Guided Tour*, defines a few relevant terms in her book *An Introduction to Genetic Algorithms*. In a genetic algorithm, the "population" is the set of solutions being tested, each determined by a sequence of numbers called a "genome." "Evaluation" is the phase during which each member of the population is tested to determine its "fitness," defined as the value of some numerical function depending on the problem being solved (in this paper, any mention of a solution's "quality" refers to its fitness). "Selection" is the process of choosing the fittest population members, "reproduction" or "recombination" is the process of combining the selected gene sequences to form offspring, and "mutation" is the process of introducing random changes into the new genomes (Mitchell, 1996, p. 5-8).

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

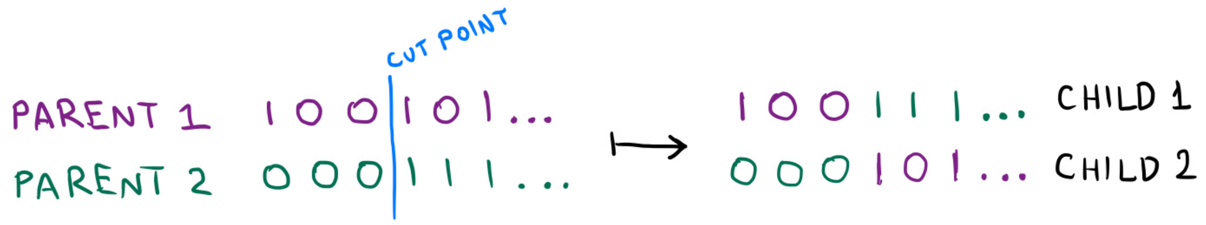
Genetic algorithms have been applied to various mathematical problems, sometimes producing solutions surpassing those designed by humans. Al-Sultan, Hussain and Nizami (1996) design a GA to find quick approximate solutions to the "set-covering problem", which is computationally time-consuming to solve 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) found that a GA outperformed four other manmade algorithmic solutions to the "bridge club scheduling problem" with respect to quality and efficiency. These findings confirm that mindless GAs can often display more mathematical "ingenuity" than humans.

GAs have also been applied to various more tangible real-world problems. Metawa, Hassan, and Elhosney (2017) apply GAs to bank lending that develop formulae which use financial data to generate optimal loan decisions. Xiao (n.d.) applies a GA to spatial partitioning problems relevant to political redistricting. Holland (1992) explains how GAs, combined with physics simulators, helped design optimal jet engine turbines. Various researchers have applied GAs to robot movement and path planning, like Achour and Chaalal (1996), Bezák (2012) and Sedreh and Zadeh (2018). GAs' plentiful applications need not be restricted to the private sector: Beligiannis, Moschopoulos and Likothanassis (2009) claim that their GA-designed method of school scheduling is a significant upgrade from the standard method due to its customizability.

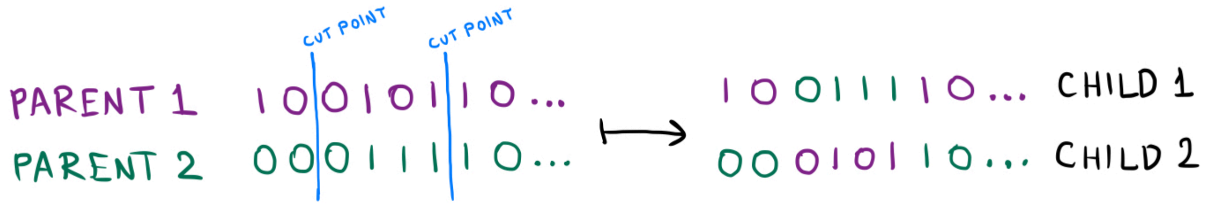
GAs are clearly powerful tools of mathematical discovery, private profit, public benefit, and technological innovation. In the next section, we discuss how tweaking a GA's structure can alter its performance. Because GAs have wide-ranging applications, any improvement in their structure could potentially enhance any of its implementations listed above.

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

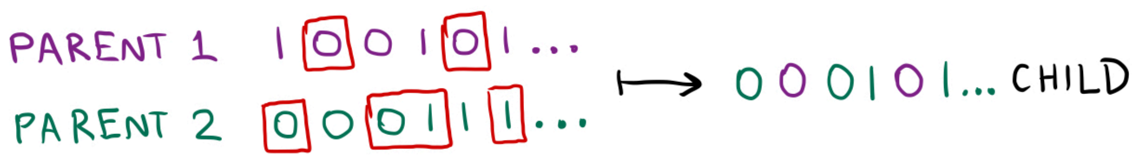
"Reproduction" was defined earlier as the process during which a population's selected genomes are combined to form a new population. However, strings of digits can be "combined" in many different ways. The genomes of two parents could be spliced at a point, like this:



...or they could be spliced using two cut points:



...or they could even be mixed together character by character, like this:



Each different method is called a "crossover operator" (CO). Umbarkar and Sheth (2015) and Kora and Yadlapalli (2017) list various crossover operators. It happens that the first crossover operator showcased above is called "single-point crossover," the second is "two-point crossover," and the third is "single point crossover" (although Umbarkar and Sheth call it "discrete crossover" when only one child is formed). Their papers include many different and more elaborate COs, some of which even involve more than two parents.

The effect of a GA's CO on its efficiency and solution quality 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 (2010a) compared the many COs' performance and remarked 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, ranking single-point crossover best and uniform crossover second-best, partially contradicting the results of Picek and Golub (2010a). These results are not necessarily incompatible, but rather suggest 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 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 COs should be ignored - various researchers have shown that they can significantly impact a 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 with three or more parents per child often outperformed standard two-parent crossover.

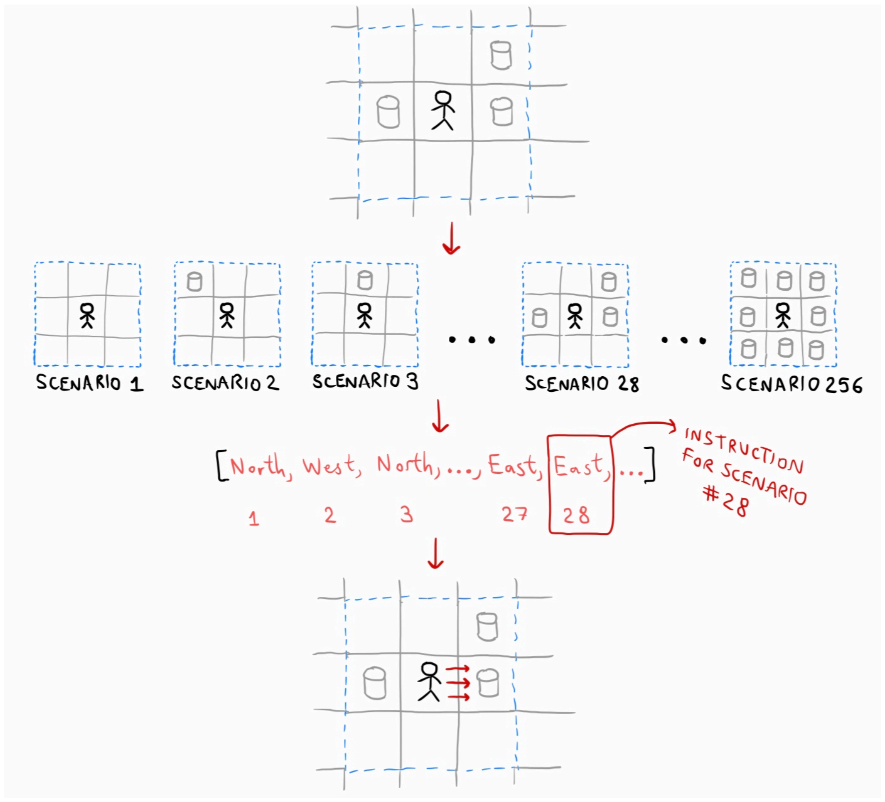
This suggests the following question: how does the CO used in a GA affect the quality of its solutions? Of course, it is impossible to answer this question in general, since the optimal CO varies depending on the problem to which the GA is applied. However, the purpose of this paper is to shed some light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

2. Methods

2.1. The Problem

In her 2009 book *Complexity: A Guided Tour*, Melanie Mitchell illustratively applies a GA to a simple problem: a game played on a 10x10 grid by a member of her virtual population (who she affectionately names Robby the soda-can-collecting Robot) in which "soda-cans" are randomly distributed on the grid and the player is awarded points for collecting as many as possible in a limited number of turns. We will apply GAs to a variation of Mitchell's "soda-can collection problem." The purpose of this research is not only to compare the effects of various COs, but also to propose intuitive explanations for these effects. Therefore, this problem's simplicity makes it an ideal choice, since the focus of this research is the structure of a GA, not the problem that it is solving.

The modified version of Mitchell's problem proceeds as follows. The game takes place on a 10x10 square grid that wraps around at the edges (that is, a 10x10 toroidal grid). For each square, a "soda can" is placed in that square with probability 0.5, randomly scattering cans across the grid. The player starts in a corner and moves about the board according to rules determined by its genome.


SCENARIO 1
SCENARIO 2
SCENARIO 3
SCENARIO 28
SCENARIO 256
[North, West, North, ..., East, East, ...]
1
2
3
27
28
INSTRUCTION FOR SCENARIO #28

During each turn, the player "observes" the 8 squares to which it is horizontally, vertically, and diagonally adjacent. Each of these squares either contains a soda can or does not, so there are $2^8 = 256$ possible observations. These scenarios are enumerated, and each is assigned an instruction (to move north, south, east, or west) in the player's gene sequence. The player follows its genomic instructions for 50 turns, at which point the game ends the player's score equals the total number of cans collected.

2.2. Implementation and Data Collection

A single trial of the GA consists of these steps:

(1) An **initial population** of 50 individuals is generated.

(2) 40 10 x 10 grids are generated and stored.

(3) **Evaluation**: each player plays the game on each grid and receives a fitness value equal to its average score.

(4) **Selection**: players are sorted by fitness value, and those in the lowest 50% are discarded.

(5) Depending on the CO used, one or more parents are chosen uniformly at random from the selected players, and their genomes are recombined to form one member of the new population. (An individual could be selected as a parent more than once, so individuals can sometimes reproduce with themselves.)

(6) **Mutation**: each digit in the child's gene sequence may randomly be changed with probability *p*, which varies depending on the trial.

(7) **Reproduction**: steps 5 and 6 are repeated until the new population is as large as the previous population.

(8) The new population replaces the old population.

(9) Steps 2-8 are repeated 100 times.

This conforms to the GA structure outlined in an earlier flowchart, derived from the process used by numerous other researchers. The decision to run the algorithm for 100 generations is based on preliminary observations that population fitness usually plateaus by generation 100. The researcher's Python implementation of this GA can be found in **Appendix A**.

These statistics are collected from a trial as it runs. The first and second statistics assess the solution quality and convergence speed of the GA, while the third and fourth help measure its genetic diversity:

- The average fitness value of each generation's population
- The fitness value of the "most fit" player from each generation's population
- The variance in fitness values of each generation's population
- The average hamming distance between genomes (defined as the number of characterwise differences between them) of each generation's population

The following is a list of all COs employed, along with descriptions and abbreviations that will be used later for brevity's sake. For visual explanations of each operator, see **Appendix B**.

1. Asexual (**AS**) - a parent duplicates itself exactly.
2. 2-parent single-point crossover (**2P1PX**) - a cut point is randomly selected, and the child's genes before the cut point come from one parent, while those after the cut point come from another parent.
3. 2-parent double-point crossover (**2P2PX**) - two cut points are randomly selected, and the child's genes between the two cut points come from one parent, while the rest come from another parent.
4. 3-parent double-point crossover (**3P2PX**) - two cut points are randomly selected, and the child's genes before the first cut point come from one parent, the genes between the cut points come from another parent, and the genes after the second cut point come from a third parent.
5. 2-parent segmented crossover (**2PSX**) - each character in the genome becomes a cut point with probability p=0.2. The child's genes come from one parent until a cut point is reached, at which point the child takes genes from a second parent, switching back to the first parent when another cut point is reached, and so on.
6. 3-parent segmented crossover (**3PSX**) - cut points are chosen as in 2PSX. The child's genes are taken from the first parent until a cut point is reached, then they are taken from the second parent until another cut point is reached, then from the third parent until another cut point is reached, and finally genes are taken from the first parent again, and this process repeats until the genome is filled.
7. 2-parent uniform crossover (**2PUX**) - each character in the child's genome is equated to the corresponding character in either the first or the second parent's genome, with equal probability p=0.5.
8. 3-parent uniform crossover (**3PUX**) - each character in the child's genome is equated to the corresponding character in either the first, second, or third parent's genome with equal probability p=0.33.
9. Wright's Heuristic 2-parent uniform crossover (**W2PUX**) - each character in the child's genome is either equated to the corresponding character in either the first or second parent's genome, with a bias towards the more fit parent. More specifically, the probability that a character is taken from a particular parent is proportional to that parent's fitness value.
10. Wright's Heuristic 3-parent uniform crossover (**W3PUX**) - analogous to W2PUX, but with three parents.
11. Universal uniform crossover (**UUX**) - all selected players become parents, and the elements of the child's gene sequence are chosen from among them with equal probability.

All of the above COs were obtained from or inspired by the literature, with minor modification by the researcher. Deslauriers (2006) compared asexual and sexual COs with very nuanced results, justifying the inclusion of **AS**. Umbarkar and Sheth (2015) and Kora and Yadlapalli (2017) list **2P1PX**, **2P2PX**, and **2PUX** as common COs, Picek and Golub (2010b) describe **2PSX**, and Lim et. al. (2017) describe Wright's Heuristic crossover, which the researcher combined with uniform crossover to create **W2PUX**. The findings of Eiben and Raué (1994) suggest that COs with more than two parents often outperform 2-parent COs, inspiring the researcher to develop **3P2PX**, **3PSX**, **3PUX**, **W3PUX**, and **UUX**, which are modified versions of previously mentioned COs.

The entirety of data collection consists of 5500 trials. Trials are partitioned into 55 groups of 100 trials each, and the trials in a particular group are determined by a specific CO and one of five mutation probabilities (MPs) p=0.005,0.01,0.03,0.05, or 0.07. Grouping the trials this way should allow the researcher to observe both the isolated effects of each CO and the compatibility of each CO with higher and lower MPs.

2.3. Data Analysis and Limitations

Data analysis is divided into two phases. During the first phase (sections **3**), the 11 COs are compared using raw data and summary statistics. The collected data is manipulated to form objective measurements of solution quality, convergence speed, and population diversity, and statistical significance are evaluated (section **3.3** and **Appendix D**). During the second phase (section **4**), the researcher attempts to intuitively explain *why* and *how* the COs affect the the GA. This section of data analysis is less rigorous because it involves the (admittedly error-prone) researcher using mathematical and evolutionary intuition to tentatively explain the phenomena observed.

Recall that the findings of Manger (2013) suggested that a CO's effect on a GA may vary from problem to problem, meaning that any raw quantitative results obtained will be specific to the "soda can collection problem" and not generalizable to GAs as a whole. Although the hypothesizing in section **4** involves some speculation, it may shed light on GAs in general, producing less reliable but more generalizable results.

That said, generalizability will still be limited. The 11 COs chosen for examination are by no means exhaustive, and there are so many conceivable COs that the researcher cannot examine all of them. Further, a whole class of GAs use genomes containing continuous decimals rather than discrete integers, making them so different from the GAs considered here that they are nearly untouchable by any generalization.

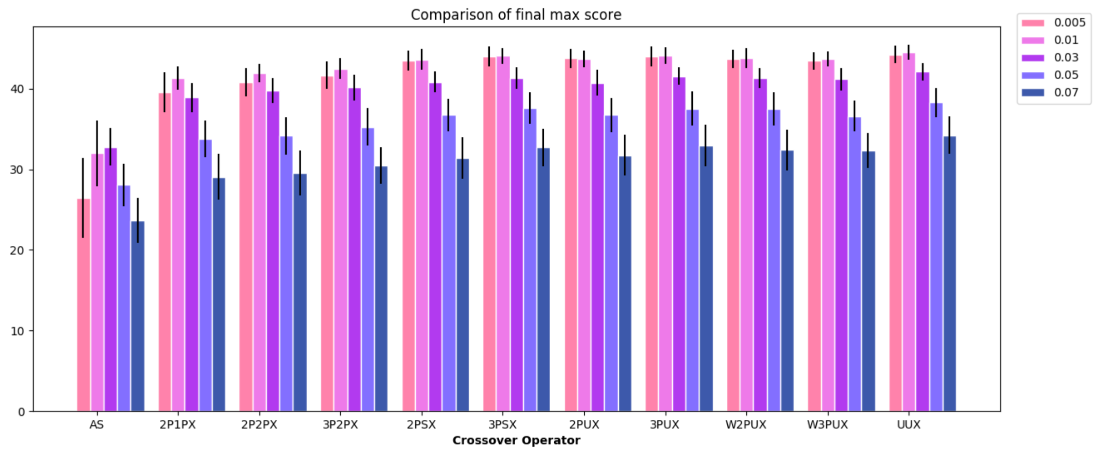
3. Data Analysis

Data collection occurred on a remote server, the data was stored in an SQL database, and the researcher used Python code to retrieve and manipulate the stored results. All data collected can be found in **Appendix C**. Matplotlib, a Python package that is useful for graphically displaying data, has been used to summarize the data. In this section, summary statistics and graphs are displayed, observations are made, and questions are posed and tentatively explained using evolutionary reasoning.

The researcher noticed during preliminary analysis that the COs **2P1PX**, **2P2PX**, and **3P2PX** exhibit similar behavior, so they are henceforth referred to as the **S1** COs for brevity's sake. The other 7 sexual COs are referred to as the **S2** COs.

3.1. Quality of Solutions and Speed of Convergence

The graph below compares the fitness of the highest-performing individual in each GA's 100th generation population, averaged over 100 trials for each of the 55 combinations of CO and MP. The error bars on top of each bar delimit plus/minus one standard deviation.

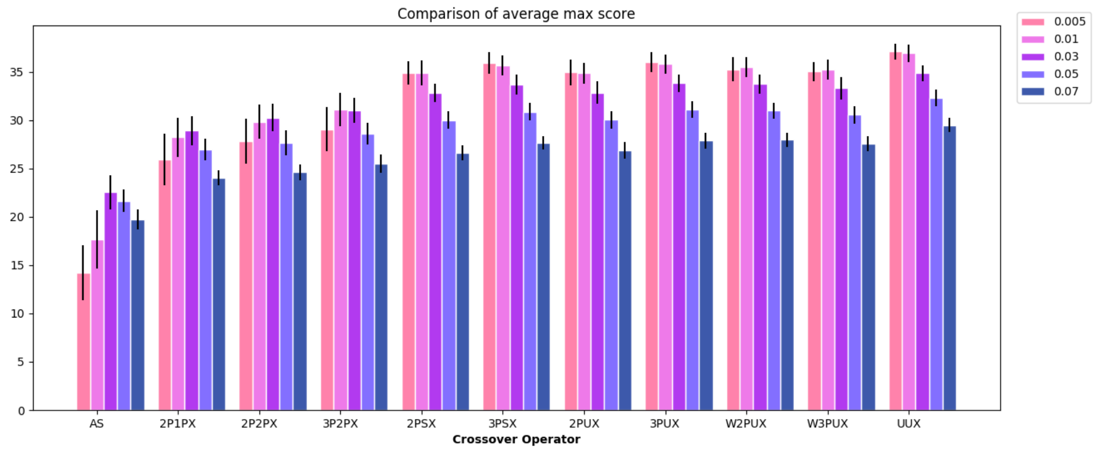


AS	2P1PX	2P2PX	3P2PX	2PSX	3PSX	2PUX	3PUX	W2PUX	W3PUX	UUX
26.439	39.5388	40.7916	41.6698	43.4712	43.9968	43.7426	43.9696	43.6734	43.448	44.2572
31.9962	41.3112	41.9296	42.5004	43.6266	44.0598	43.6688	44.0826	43.8188	43.65	44.5074
32.7772	38.9094	39.7598	40.1436	40.848	41.2892	40.7356	41.5336	41.3224	41.1764	42.1068
28.046	33.7782	34.1526	35.2556	36.7266	37.582	36.7354	37.5346	37.4582	36.6126	38.269
23.6404	29.0736	29.5042	30.4654	31.3948	32.6904	31.7174	32.9372	32.3866	32.3448	34.2318

Despite being surprisingly regular, this data suggests a few questions. Why...

- does **AS** perform significantly worse than the others?
- do the **S1** COs perform much better than **AS** but slightly worse than the **S2** COs?
- does **AS** perform best at a higher MP than the other COs?
- do the **S1** COs perform best at a higher MP than **AS** but a lower MP than the **S2** COs?
- does **UUX** always perform best?
- does **W3PUX** underperform **3PUX**, despite being biased to help higher-scoring players distribute their genes more?

Now we consider the average of the top fitness values in each of each GA's 100 populations, rather than just the top score in the 100th generation:



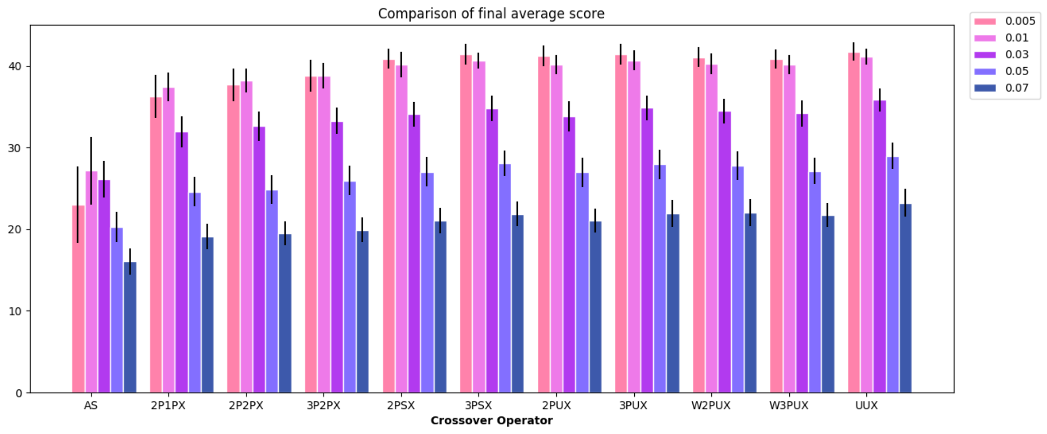
AS	2P1PX	2P2PX	3P2PX	2PSX	3PSX	2PUX	3PUX	W2PUX	W3PUX	UUX
14.21203	25.96666	27.85645	29.08889	34.90739	35.95793	34.97573	36.01934	35.27862	35.04209	37.12225
17.63887	28.239	29.8596	31.12302	34.87811	35.67236	34.8693	35.81695	35.49718	35.26543	36.94087
22.58074	28.92246	30.29402	31.02219	32.81525	33.71996	32.8817	33.838	33.75217	33.31983	34.87358
21.66058	26.97834	27.66324	28.6261	30.02391	30.88401	30.04706	31.11396	31.00756	30.59177	32.34193
19.73564	24.00825	24.63183	25.51201	26.6634	27.6642	26.894	27.91633	27.97332	27.58169	29.49695

This data exaggerates many of the phenomena observed in the previous graph, such as the relative success of **UUX** and failure of **AS** and the **S1** COs. Wright's Heuristic also fails again: **W2PUX** barely outperforms **2PUX** but **W3PUX** underperforms **3PUX**.

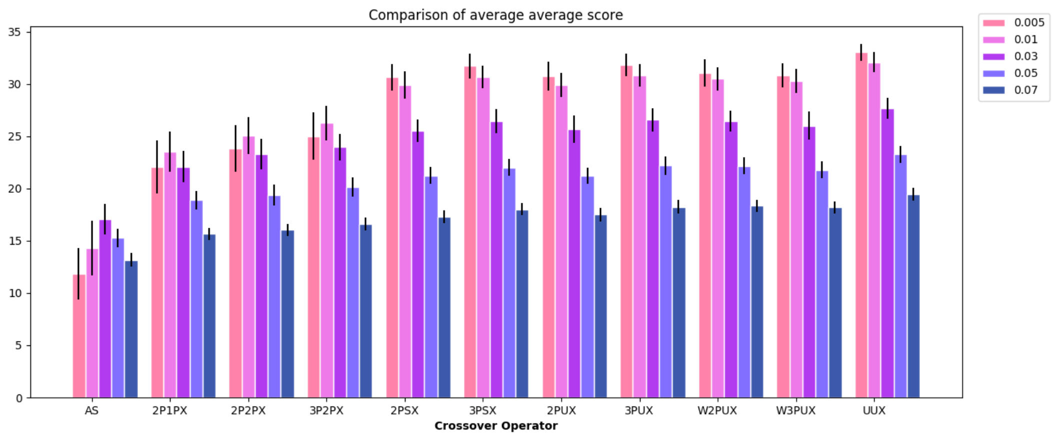
Because this data measures maximum fitness values averaged over all generations of each GA, it also has implications on convergence speed. A higher score here indicates not only that a GA produced high-performing solutions, but also that obtained such solutions sooner. Because the **S2** COs outperform **AS** and the **S1** COs by a greater margin than before, we might infer that they not only yield better solutions, but also do so more quickly. This suggests the question:

- Why do **AS** and **S1** COs (presumably) produce good solutions more slowly than **S2** COs?

Now we turn to the data regarding scores averaged over all individuals in a GA's population. The first data set below summarizes the average population score for each GA's 100th generation population, and the second data set summarizes the average population score averaged over all 100 generations of each GA:



AS	2P1PX	2P2PX	3P2PX	2PSX	3PSX	2PUX	3PUX	W2PUX	W3PUX	UUX
22.99582	36.22926	37.66614	38.74915	40.86544	41.45294	41.24358	41.43475	41.06741	40.83691	41.75411
27.14413	37.42072	38.23246	38.77306	40.11938	40.61315	40.12787	40.6649	40.24395	40.13844	41.12356
26.10199	31.92566	32.61193	33.25196	34.06406	34.77908	33.80488	34.84787	34.49054	34.1644	35.81642
20.25139	24.58113	24.82574	25.95681	27.02813	28.08104	26.98124	27.92054	27.76348	27.137	28.96909
16.06355	19.09896	19.48903	19.92572	21.0676	21.85412	21.07076	21.90792	21.9869	21.73905	23.23704



AS	2P1PX	2P2PX	3P2PX	2PSX	3PSX	2PUX	3PUX	W2PUX	W3PUX	UUX
11.83435	22.07162	23.81274	25.00605	30.63375	31.73569	30.75546	31.82305	31.05157	30.83306	33.0188
14.26565	23.52826	25.06323	26.25182	29.90125	30.67902	29.89034	30.84402	30.49961	30.26951	32.06316
17.04276	22.09651	23.29665	23.96697	25.50519	26.46172	25.67812	26.57467	26.45925	26.0115	27.68615
15.27696	18.87195	19.37067	20.13914	21.23628	22.01425	21.20264	22.18298	22.16601	21.78227	23.28604
13.15376	15.64413	16.03261	16.60505	17.32297	18.02034	17.49179	18.24214	18.33601	18.19902	19.42991

Again, we see a marked distinction between **AS**, **S1** COs, and **S2** COs, with only slight discrepancies within the **S2** group. Most of the notable phenomena observed earlier are also present here. However, it appears that most COs now exhibit much lower performance at higher MPs. This might lead one to ask

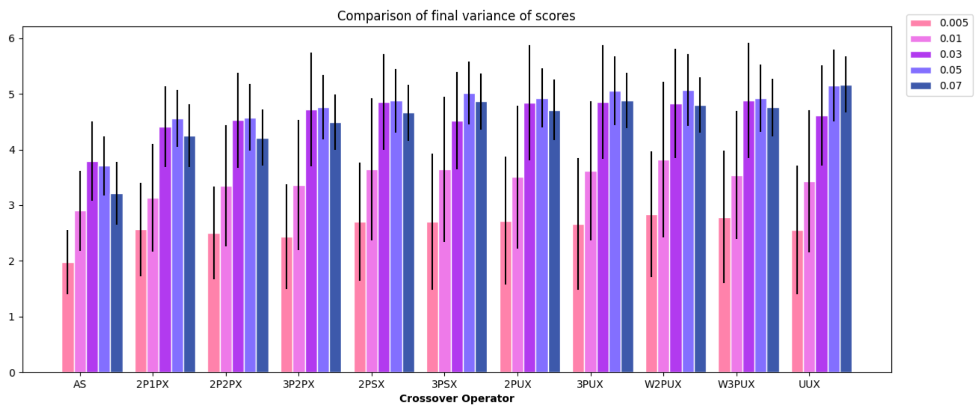
- Why do most COs' performance decrease much more rapidly at higher MPs when considering average score than when considering maximum score?

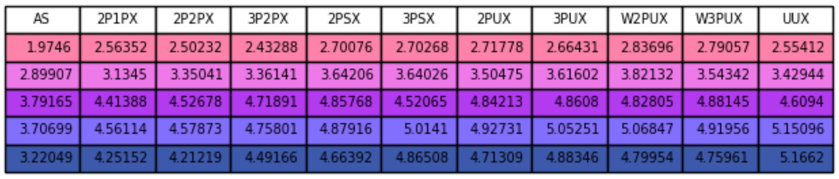
Having considered all data regarding GA performance and convergence speed, we now pose a few general questions about phenomena that reappear regardless of the measurement used. Why...

- does the **S2** group outperform the **S1** group, which in turn outperforms **AS**?
- does **AS** seem to perform best at the highest MP, the **S1** COs at a slightly lower MP, and the **S2** COs at the lowest MP?
- does **UUX** consistently outperform other COs?
- does **3PUX** outperform its Wright's Heuristic variant **W3PUX**?

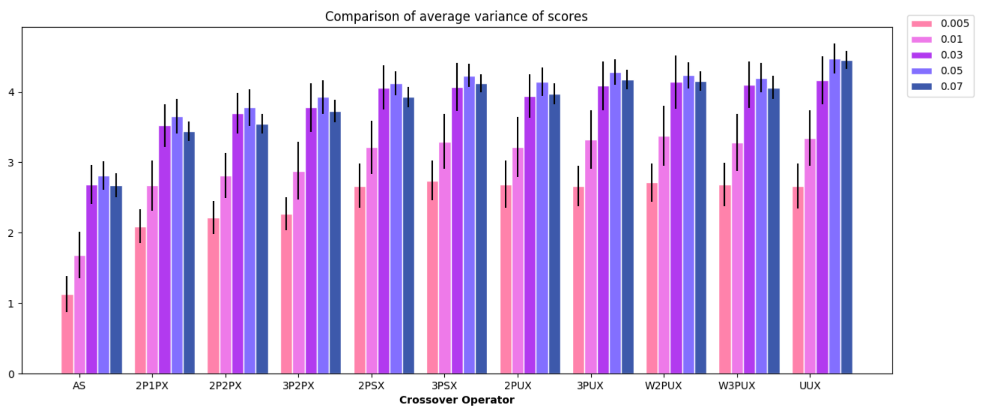
3.2. Population Diversity

Next, we briefly consider the different COs' effects on a GA's population diversity. As mentioned earlier, two different measurements are used: variance in population fitness values, and average hamming distance between population genomes. The two data sets below summarize the variance in population fitness values (the first draws only from the final generations of each GA, and the second averages variances from all 100 generations):





AS	2P1PX	2P2PX	3P2PX	2PSX	3PSX	2PUX	3PUX	W2PUX	W3PUX	UUX
1.9746	2.56352	2.50232	2.43288	2.70076	2.70268	2.71778	2.66431	2.83696	2.79057	2.55412
2.89907	3.1345	3.35041	3.36141	3.64206	3.64026	3.50475	3.61602	3.82132	3.54342	3.42944
3.79165	4.41388	4.52678	4.71891	4.85768	4.52065	4.84213	4.8608	4.82805	4.88145	4.6094
3.70699	4.56114	4.57873	4.75801	4.87916	5.0141	4.92731	5.05251	5.06847	4.91956	5.15096
3.22049	4.25152	4.21219	4.49166	4.66392	4.86508	4.71309	4.88346	4.79954	4.75961	5.1662

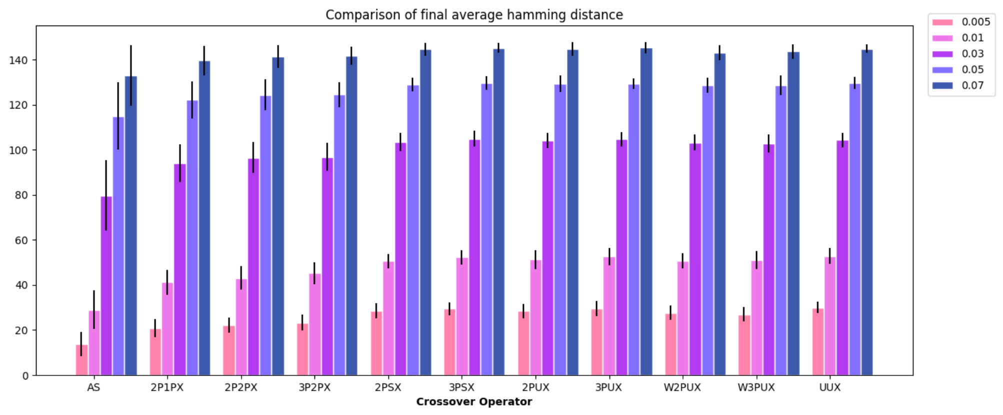


AS	2P1PX	2P2PX	3P2PX	2PSX	3PSX	2PUX	3PUX	W2PUX	W3PUX	UUX
1.12909	2.08941	2.21168	2.26855	2.66825	2.74255	2.68796	2.66598	2.71337	2.68347	2.6597
1.68427	2.67	2.8126	2.88104	3.21306	3.29443	3.21609	3.32613	3.3776	3.28373	3.34799
2.68509	3.52262	3.69462	3.77749	4.06097	4.0653	3.94014	4.08594	4.14196	4.09773	4.17006
2.80933	3.65323	3.77795	3.93039	4.1229	4.23484	4.14539	4.28234	4.23638	4.20188	4.47583
2.67428	3.44405	3.54949	3.72689	3.92609	4.12391	3.97461	4.17471	4.15455	4.06108	4.4566

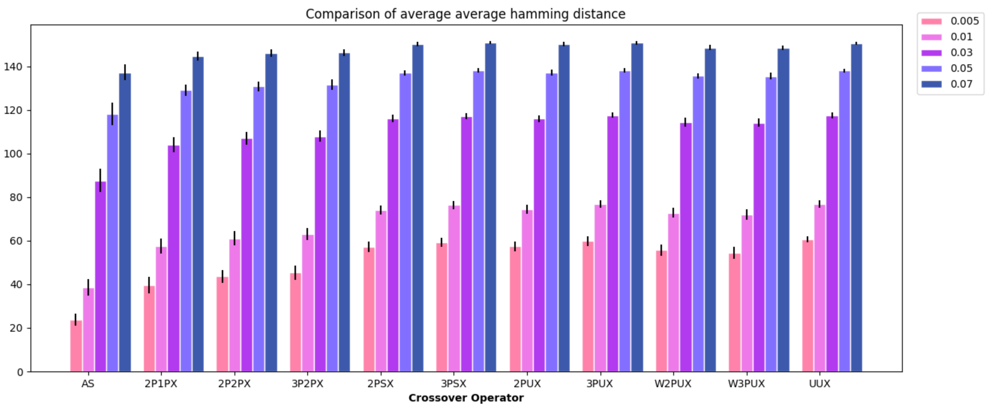
Remarkably, the same pattern as before appears: **AS** has the lowest variance, the **S1** COs have higher variance, and the **S2** COs have a slightly higher variance. Aside from this ranking, these graphs display no other apparent noteworthy characteristics. In fact, all COs behave very similarly for corresponding MP values. This suggests only one question:

- Why do **AS** populations tend to have the lowest score variance, **S1** populations greater score variance, and **S2** the greatest score variance?

The following graphs and data tables summarize average hamming distance between pairs of individuals in populations (again, the first data set is taken only from the final generation of each GA, while the second is averaged over all generations):



AS	2P1PX	2P2PX	3P2PX	2PSX	3PSX	2PUX	3PUX	W2PUX	W3PUX	UUX
13.80871	20.83077	22.22994	23.28303	28.51032	29.43335	28.42689	29.56498	27.71443	26.90389	30.03839
28.99998	41.24073	43.1492	45.29424	50.58411	52.29497	51.2514	52.60565	50.72332	51.16384	52.86597
79.75484	94.24051	96.55265	96.86953	103.53683	104.92552	104.14277	104.71131	103.22888	102.86772	104.39892
115.0009	122.23887	124.46957	124.55156	129.05836	129.74337	129.2571	129.24927	128.60768	128.56961	129.84188
133.01033	139.68201	141.53304	141.73604	144.73015	145.21739	144.82851	145.35633	143.21352	143.71951	145.01225



AS	2P1PX	2P2PX	3P2PX	2PSX	3PSX	2PUX	3PUX	W2PUX	W3PUX	UUX
23.74606	39.67194	43.61925	45.37588	57.22146	59.33894	57.49397	59.8102	55.65766	54.49486	60.55321
38.62811	57.46377	61.08993	63.05614	74.15567	76.35019	74.48773	76.7719	72.80716	72.01251	76.90259
87.63293	104.03111	106.99974	107.97305	115.98633	117.18464	115.95623	117.59966	114.42852	114.21681	117.53183
118.26633	129.09475	130.88205	131.65946	137.02939	138.10723	137.19286	138.15005	135.6236	135.59123	138.01243
137.16916	144.72833	146.08747	146.32367	150.13813	150.84145	150.17182	150.78207	148.6477	148.61812	150.55612

The same ordering has reappeared again - **AS** is conducive to the lowest hamming distance, **S1** COs slightly higher hamming distances, and **S2** COs the highest average hamming distance. Thus, we have the question:

- Why do **AS**, the **S1** group, and the **S2** group give rise to increasingly average hamming distances?

3.3. Statistical Confidence:

The researcher's statistical confidence in each of the following observations has been evaluated:

- **S1** COs outperform **AS**, and **S2** COs outperform **S1** COs
- **AS** performs best at a higher MP than **S1** COs, and **S1** COs perform best at a higher MP than **S2** COs (for most measurements)
- **UUX** performs best
- **3PUX** outperforms its Wright's Heuristic counterpart **W3PUX**
- **S1** COs have a higher score variance than **AS**, and **S2** COs have a higher score variance than **S1** COs

To calculate p-values pertaining to each of these observations, the researcher used traditional methods of statistical hypothesis testing as described in the Handbook of Biological Statistics written by J.H. McDonald, a professor in the University of Delaware's Department of Biological Sciences. Although this research is not technically biological, much of the statistical hypothesis testing used in biology pertains to analyzing populations and genetics, which this research simulates.

However, the researcher will refrain from "accepting" or "rejecting" any of the above claims on the basis of the p-values. The task of translating p-values into confidence or reliability shall be left to the reader. This is because, in recent years, conventional best practices of statistical significance have been called into question by the scientific community, as Benjamin et. al. (2017), Amrhein, Greenland, and McShane (2019), and Beck (2016) discuss.

Furthermore, because of the many different COs, MPs, and measurements of performance and variability, *many* different p-values must be computed. Presenting the values here would be cumbersome, so tables of relevant values have been numbered and stored in **Appendix D**. For each comparison, the null hypothesis is taken to be the claim that the two quantities in question are equal (the hypothesis tests are one-tailed). Since many p-values are incredibly small (some on the order of 10^{-100}), they have been stored as z-scores of a normal distribution instead of p-values. **Appendix D** also contains a table of cumulative normal distribution values (calculated using online Wolfram Alpha software) so that the reader can translate these z-scores into p-values.

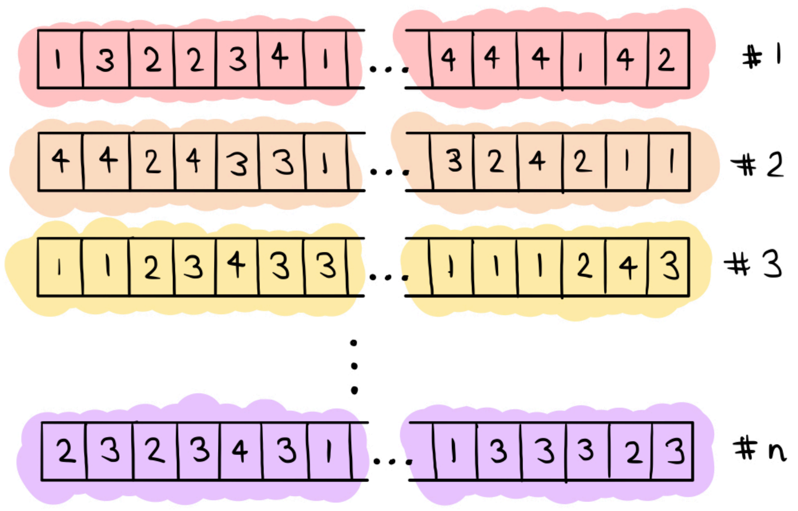
4. Discussion and Conclusions

4.1. Tentative Explanations

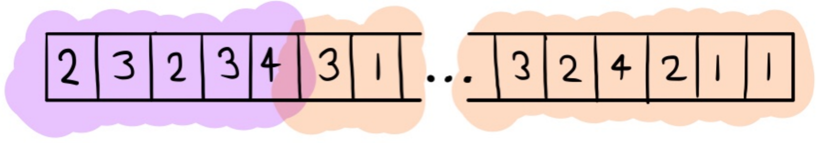
To intuitively explain the results observed, we first remark on a connection between GA performance and population diversity. Because the solution space (the set of all possible solutions to the problem, or in this case the set of all possible genomes) is enormous - for the "soda can collection problem," there are $4^{256} = 1.34 \times 10^{154}$ different strategies - a GA must sift through many possibilities before finding high-quality solutions. This suggests that a GA's population diversity should be somewhat positively correlated with its performance. The data confirms this hypothesis: recall that the relative CO rankings with respect to performance and diversity were very similar (in both cases, **AS** ranked lowest, **S2** COs ranked highest, and **S1** COs in between).

This explains why the same COs displayed high performance and high diversity. However, it does not explain why particular COs displayed high/low performance and diversity, which is the question we consider next. Why were performance and diversity lowest for **AS**, slightly greater for **S1** COs, and greatest for **S2** COs (and absolutely greatest for **UUX**)?

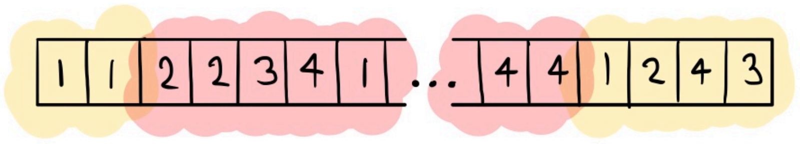
The answer seems to be that the increasingly complex mechanisms of **AS**, **S1**, and **S2** allow them to be increasingly "creative" while generating offspring - that is, **S2** COs can produce more possible different offspring genomes than **S1** COs, and **AS** produces the fewest. To see why, consider the following heuristic mathematical model. Suppose we have a populatiom of n selected individuals, and their gene sequences are "stained" different colors as shown below:



Suppose these individuals undergo reproduction and each character in the child genome retains the coloration of the genome it came from. If **AS** is used, the child's genome must be monochromatic and can be one of n different colors (since asexually produced children are exact copies of their parents). Thus, **AS** can produce n different offspring coloration patterns. However, **2P1PX** can produce child genomes that look like this, which **AS** cannot produce:



Using basic combinatorics, we can calculate the number of different possible coloration patterns produced by **2P1PX** for a 256-character child genome as exactly $255n^2 - 254n$. This is *much* more than **AS** can produce! However, **2P2PX** yields even more different patterns, some of which neither **AS** nor **2P1PX** can produce, such as the one shown below:



Another combinatorial calculation shows that **2P2PX** can give rise to exactly $32640n^2 - 255n$ coloration patterns. The table below contains exact formulas for the number of different offspring coloration patterns that each CO produces, as well as the approximate number of patterns for n=50, the population size used in our GA trials:

	Formula in terms of n	**Value for n = 50**
AS	n	50
2P1PX	$255n^2 - 254n$	6.248×10^5
2P2PX	$32640n^2 - 32639n$	7.996×10^7
3P2PX	$32640n^3 - 65280n^2 + 32641n$	3.918×10^9
2PSX	$2^{255}n^2 - (2^{255} - 1)n$	1.418×10^{80}
3PSX	$2^{255}n^3 - 2^{256}n^2 + (2^{255} + 1)n$	6.950×10^{81}
2PUX	$2^{255}n^2 - (2^{255} - 1)n$	1.418×10^{80}
3PUX	$(3^{255}/2)n^3 - (3^{256}/2 - 2^{255})n^2 + (3^{255} - 2^{255} + 1)n$	2.840×10^{126}
W2PUX	$2^{255}n^2 - (2^{255} - 1)n$	1.418×10^{80}
W3PUX	$(3^{255}/2)n^3 - (3^{256}/2 - 2^{255})n^2 + (3^{255} - 2^{255} + 1)n$	2.840×10^{126}
UUX	n^{256}	8.636×10^{434}

Although this is only a rough metric of each GA's conduciveness to diversity, it does clearly separate the COs by intuitively demonstrating how some are clearly more "creative" than others. It matches up nicely with our observations: **AS** produces fewer than 100 coloration patterns, **S1** COs produce on the order of 10^5 - 10^{10}, **S2** COs produce on the order of 10^{80}-10^{130}, and **UUX**, the most successful and diverse CO, produces over 10^{434}.

The correlation between a CO's "creativity" and its performance might also explain two other noteworthy phenomena: the higher performance of **AS** at greater MPs, and the failure of Wright's Heuristic to significantly improve performance of **2PUX** and **3PUX**. Because the reproduction mechanism of **AS** fails to shuffle genes in a combinatorially rich way like the other COs, mutation is the only source of new genetic variability. In sexual COs, mutation might interfere by sabotaging high-quality solutions (the data supports this hypothesis - performance drops sharply at higher MPs in sexual COs), but in **AS**, mutation is the only mechanism by which novel solutions appear. Regarding Wright's Heuristic: note that the different coloration patterns produced by **2PUX** and **3PUX** are equally likely, and although **W2PUX** and **W3PUX** have the same number of possible patterns as their counterparts, the outcomes have unequal probability because they are biased in favor of one parent, meaning that Wright's Heuristic potentially *reduces* a CO's likelihood of creating novel solutions.

To summarize the researcher's conjectures:

- Population performance and diversity go hand in hand, since variability increases a GA's likelihood of developing a high-quality solution by chance

- The “gene coloration” model described above supports the following ranking of COs in order of increasing performance and diversity, as manifested in the data: **AS**, **S1**, **S2**, **UUX**
- A key feature of successful COs is their “creativity,” or ability to produce novel solutions
- If a CO is not “creative” enough, mutation is the predominant source of novelty, as appears to be the case with **AS**
- Wright’s Heuristic seems to slightly decrease diversity, thereby decreasing “creativity” and possibly overall performance

4.2. Limitations and Thoughts for Further Research

To list some limitations of this research:

- The above explanations come from the researcher’s own intuition and reasoning, which may be prone to error.
- The researcher may have overlooked bugs in the code that could have skewed the data.
- Previous research suggests that GA performance can vary significantly from one problem to the next, meaning that all data and rankings obtained pertain only to the “soda can collection problem.” This limitation could be quite significant, since this problem is much simpler than many to which GAs are applied.
- Depending on the reader’s standards for statistical significance, the data trends observed may not be conclusive enough.

It is the researcher’s hope that, despite these limitations, the observations made and methods used to evaluate and explain them will be of use to future researchers and enthusiasts who strive to better understand how GAs function.

During early phases of research, the researcher intended to include an additional phase of qualitative analysis that involved comparing the movement patterns of individual population members, sampled from different generations of GAs, on the 10x10 gameboard. Due to paper length constraints imposed by College Board, this section was omitted, but it could produce elucidating insights if included in future research.

It is also worth noting that the Building Block Hypothesis, proposed by Daniel Goldberg, an influential GA researcher, seems to contradict the results obtained. This hypothesis posits that GAs succeed by accumulating and recombining small “building blocks,” or advantageous groups of genes, which remain intact (*Why genetic algorithms work,* 1997). Therefore, a highly “creative” CO such as **UUX** might actually decrease performance by scrambling successful building blocks. However, our data showed that highly creative COs performed best. One possible explanation for this is that the “soda can collection problem” is so simple that building blocks do not play a significant role. Thus, in future research, it may be advantageous to compare COs’ performance on a more complex problem.

References

Al-Sultan K.S., Hussain M.F., & Nizami, J.H. (1996). A genetic algorithm for the set covering problem. *The Journal of the Operational Research Society, 47*(5), 702-709.

Amrhein, V., Greenland, S., & McShane, B. (2019). Scientists rise up against statistical significance. *Nature.* https://www.nature.com/articles/d41586-019-00857-9.

Battarra, M., Golden, B., & Vigo, D. (2008).Tuning a parametric Clarke-Wright heuristic via a genetic algorithm. *The Journal of the Operational Research Society, 59*(11), 1568-1572.

Beck, D.L. (2016). Cover story: The fading bright line of p<0.05. *American College of Cardiology.* https://www.acc.org/latest-in-cardiology/articles/2016/05/05/13/54/cover-story.

Beligiannis, G.N., Moschopoulos, C., & Likothanassis S.D. (2009). A genetic algorithm approach to school timetabling. *The Journal of the Operational Research Society, 60*(1), 23-42.

Benjamin, D.J. et. al. (2017). Redefine statistical significance. *Nature*. Retrieved from https://www.nature.com/articles/s41562-017-0189-z.epdf.

Darwin, C. (1859). The origin of species. In P. Appleman (Ed.), *Darwin* (3rd ed., pp. 95-174). New York, NY: W.W. Norton & Co.

Deslauriers, W.A. (2006). *Asexual versus sexual reproduction in genetic algorithms*. Retrieved from Carleton University Institute of Cognitive Science website: https://carleton.ca/ics/research/technical-reports/view-reports/

Eiben, A.E., Raué, P.E., & Ruttkay Z. (1994). Genetic algorithms with multi-parent recombination. *Proceedings of the International Conference on Evolutionary Computation*, 78-87.

Emmanouilidis, C. & Hunter, A. (2000). A comparison of crossover operators in neural network feature selection with multiobjective evolutionary algorithms, GECCO-2000 workshop on evolutionary eomputation in the development of artificial neural networks, Las Vegas, 2000.

Gould, J.L., Keeton, W.T., & Gould, C.G. (1996) How natural selection operates. In P. Appleman (Ed.), *Darwin* (pp. 373-376). New York, NY: W.W. Norton & Co.

Grant, P.R. (1991). Natural selection and Darwin's finches. *Scientific American, 265*(4), 82-87.

Holland, J.H. (1992). Genetic algorithms. *Scientific American, 267*(1), 66-72.

Kora, P. & Yadlapalli, P. (2017). Crossover operators in genetic algorithms: A review. *International Journal of Computer Applications, 162*(10), 34-36.

Kumar, S.G. & Panneerselvam, R. (2017). A study of crossover operators for genetic algorithms to solve VRP and its variants and new sinusoidal motion crossover operator. *International Journal of Computational Intelligence Research, 13*(7), 1717-1733.

Lim, S., Sultan, A., Suleiman, N., Mustapha, A., & Leong, K. (2017). Crossover and mutation operators of genetic algorithms. *International journal of machine learning and computing, 7*(1), 9-12.

Magalhaes-Mendes, J. (2013). A comparative study of crossover operators for genetic algorithms to solve the job shop scheduling problem. *WSEAS Transactions on Computers, 12*(4), 164-173.

McDonald, J. H. (2014). Handbook of biological statistics. Retrieved from http://www.biostathandbook.com/index.html.

Metawa, N., Hassan M., & Elhoseny, M. (2017). Genetic algorithm based model for optimizing bank lending decisions. *Expert Systems with Applications, 80*(C), 75-82.

Mitchell, M., Crutchfield, J.P., & Das, R. (1996). Evolving cellular automata with genetic algorithms: a review of recent work. In *Proceedings of the First International Conference on Evolutionary Computation and its Applications*. Moscow, Russia: Russian Academy of Sciences.

Ortiz-Boyer, D., Hervás-Martínez, C., & García-Padrajas, N. (2005). A crossover operator for evolutionary algorithms based on population features. *Journal of Artificial Intelligence Research, 24*, 1-48.

Picek, S. & Golub, M. (2010). Comparison of a crossover operator in binary-coded genetic algorithms. *WSEAS Transactions on Computers, 9*(9), 1064-1073.

Picek, S. & Golub, M. (2010). On the efficiency of crossover operators in genetic algorithms with binary representation. In V. Munteanu, R. Raducanu, G. Dutica, A. Croitoru, V. Balas, and A. Gavrilut (Eds.), *Proceedings of the 11th WSEAS international conference on neural networks and 11th WSEAS international conference on evolutionary computing and 11th WSEAS international conference on Fuzzy systems* (pp. 167-172). Stevens Point, Wisconsin: World Scientific and Engineering Academy and Society.

Puljic, K. & Manger, R. (2013). Comparison of eight evolutionary crossover operators for the vehicle routing problem. *Mathematical Communications, 18*(2), 359-375.
Sedreh, Z. & Zadeh, M.S. (2018). Robot path planning using cellular automata and genetic algorithm. *Journal of Advances in Computer Engineering and Technology, 5*(1), 19-26.

Why genetic algorithms work. (1997, Sept 14). Genetic Algorithms. Retrieved February 20, 2020 from https://cs.stanford.edu/people/eroberts/courses/soco/projects/1997-98/genetic-algorithms/index.html.

Umbarkar, A.J. & Sheth, P.D. (2015). Crossover operators in genetic algorithms: A review. *ICTACT Journal on Soft Computing, 6*(1), 1083-1092.