Addressing SES Segregation in Charlottle-Meckelenburg Schools: A Genetic Programming Approach

Michael Robertson
{mirobertson}@davidson.edu
Davidson College
Davidson, NC 28035
U.S.A.

Abstract

Lena Parker's (Davidson College class of 2017) thesis addressed the problem of reducing socioeconomic segregation in Charlottle-Meckelenburg Schools without unreasonably increasing student commute times. Her linear programming approach struggled to run in a reasonable amount of time given the size of the problem, so we attempt to find a more efficient solution through genetic programming. Over 80 hours, our algorithm evolved solutions which improved upon the baseline fitness of random assignments by over 90%. We hope the assignments we produced would compete with and perhaps improve upon Lena's final solution. Though our results require further analysis and verification before potential presentation to the CMS (Charlotte-Mecklenburg Schools), the large relative improvement of our solutions in a comparatively short period of time demonstrates the promise of genetic programming in this problem space.

1 Introduction

The CMS schools district contained 154,434 students in 2014, and was the 18th largest district out of 121 in America (National Center for Education Statistics 2016). The mathematical problem of desegregating schools by SES (*Socio**E**conomic **S**tatus*) could be simply solved by assigning equal numbers of children of each SES level to each school. However, as Grundy describes in her 2017 book *Color and Character*, when the U.S. Supreme Court declined to hear an appeal to *Cappachione v. Charlotte-Mecklenburg Board of Education* in April 2002, it effectively upheld the ruling of two lower courts that "prior vestiges of racial discrimination had been eliminated to the extent practicable," and released the CMS from the obligation to racially desegrate placed upon it by the landmark 1971 U.S. Supreme Court case *Swann v. Charlotte-Mecklenburg Board of Education* (Grundy 2017). Later in 2002, the CMS implemented a "School Choice Plan" which led to the near immediate resegregation of schools, and they have only become "increasingly racially and socioeconomically segregated" since (Grundy 2017) (Parker 2017).

Though discussions surrounding school assignment in Charlotte have traditionally focused on racial segregation because of America's history of racial discrimination, we will here focus on SES as the segregating factor, due to the availability of student SES data provided to us through Lena's analysis and enrichment of CMS data, and because of the close correlation of race and SES in Charlotte.

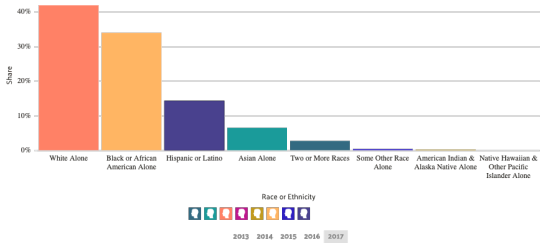

Figure 1: Population Demographics of Charlotte from 2017 Census data. Graphic created by Data USA.

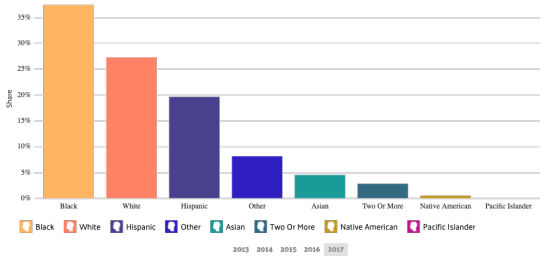

Figure 2: Each race's share of the Charlotte population living in poverty, as a percentage. Graphic created by Data USA.

Figure 1 shows that although 1.23 times more white people live in Charlotte than African American people, more than 35% of people living in poverty were African American, while less than 28% were white. Data provided by . Data USA[1] used data from the Census Bureau 2017 American Community Survey 5-year Estimate to create the visualizations above.

The difficulty of breaking SES segregation in schools comes from the clustering of populations by SES in Charlotte, as shown in figure 4, because a completely SES-equitable assignment would require unreasonable commute times. By borrowing Lena's objective function and using it as a fitness function for our genetic programming approach,

[1] datausa.io/profile/geo/charlotte-nc/#demographics, accessed 17 May 2019.

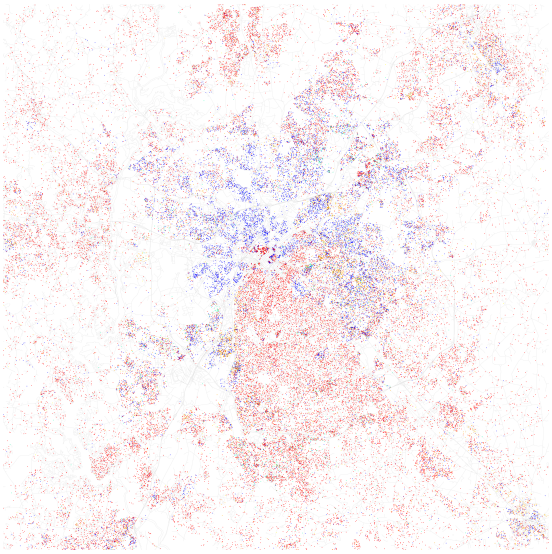

Figure 3: Geographic visualization of racial demographics in Charlotte from Census 2010 data using OpenStreetMap software, from Eric Fischer at `flickr.com/photos/walkingsf/5559889573/`, accessed 19 May 2019. Each dot represents 25 people. Red dots represent white people, blue represent black people, green Asian, orange Hispanic, and yellow other.

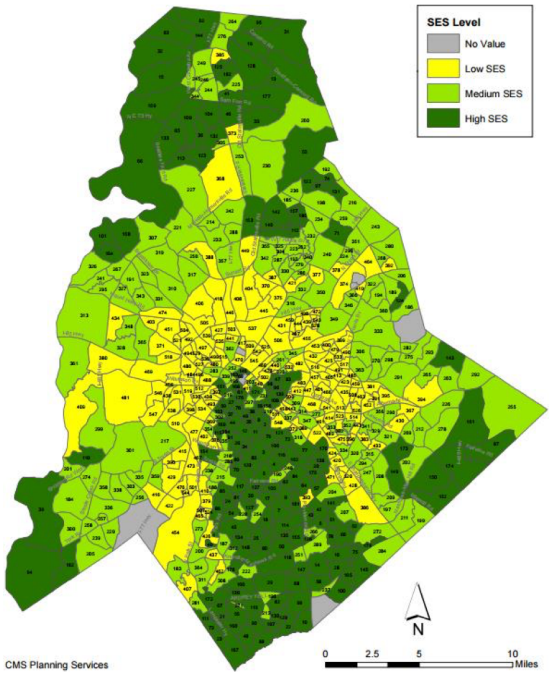

Figure 4: SES evaluation by CMS planning services in 2016 using Census Data. Though the number of dots in figure 3 obscures the underlying roads and makes visualization of the layout of Charlotte difficult, note the overlap between blue dots representing the black population in figure 3 and the yellow tracts representing low SES census tracts in figure 4. This distribution is colloquailly known as the "wedge and crescent", and appears frequently in data visualizations of Charlotte's population.

we attempt to minimize it, and return a solution of comparable or better fitness/objective function value in a shorter running time. We will outline the nature of the data Lena made available to us from her thesis work and describe our implementation of genetic programming mechanisms in the Background, and continue on to describe our experiments and analyze their results.

2 Background

For her senior thesis Lena collected and estimated data points for each of the 47,150 roads in the Open Mapping Mecklenburg dataset and for each of the 94 schools. We used nearly all of the data she provided, and though we invested a significant amount of time in data preparation, we would have accomplished nothing in our given time frame had not Lena given us the data in a compact and naturally intelligible format, and readily assisted in helping us interpret it.

Data

We received two data files from Lena which partitioned the CMS district into "bluegreen" and "greyvoilet" sections. We combined these sets to form a dataset that encompassed the entire CMS school district.

Given Data Each of these files provided well-organized data we which we preprocessed into our model of the problem. We used four items of data from each of the .csv files:

- **Road Populations** By merging census tract population data with a shape file of all Mecklenburg Country roads from Open Mapping Mecklenburg[2], Lena estimated the population of five to nine-year-old children in each individual road segment based on the proportion of the road to the total road length in the census tract and the population in that tract. For example, if road number five had length 1 mile in a census tract with 1,000 five to nine-year-olds and 10 miles of road in it, then she would estimate that road to have $(1000)\frac{1}{10} = 100$ children in it.
- **Road SES Status** Lena determined a road's SES status by assigning it the SES status of the census tract the majority of it fell within. Lena then considered the estimated population living in that road as the SES of that road. Expanding upon our previous example, if road five lay in a low SES census tract, all ten people in it would be considered Low SES.
- **School Capacities** Lena reported the CMS's stated capacities for each of the 94 schools in the district.
- **Commuting Times** Using the locations of the 94 schools and speed limits on road segments from Open Mapping Mecklenburg, Lena estimated commute times from each road to each school and reported the commute times to the five closest schools. She reasoned that no assignment should require a student to attend a school further away from their home than five other schools.

[2] maps.co.mecklenburg.nc.us/openmapping/

Data Preparation Lena formatted the data for the purpose of solving the problem using MATLAB's linear programming solving tools. We choose to represent a solution the problem as a list with length equal to the number of roads in the problem. An assignment or solution then consisted of filling each of the entries in this list with a school ID. So, in a hypothetical problem with ten roads and five schools (with IDs 1-5), one possible solution would be:

$$[4, 5, 5, 3, 1, 5, 4, 3, 3, 4]$$

In this assignment, the children living on road number 5 will attend school 1, no one will attend school 2, the children on roads 3, 7, and 8 will attend school 3, and the children on roads 0, 6, and 9 will attend school 4. For our algorithm, we created the following structures:

- **Road Class** By reading in data from Lena's csv file, we created road objects for each road, which contained the population of the road, its SES, and a list of the five closest schools, each of which we stored as a tuple of the school ID and the distance to the school from the road. An example road might have population ten, Medium SES, and be one mile away from school one, two miles away from school two, three miles from school three, four from school four, and five miles away from school five.
- **School Class**[3] We read in a school object for each school in the given file. Each instance recorded only the capacity during data intake, but also maintains variables initialized to zero for the number of low, medium, and high SES students attending it, as well as the overall weight it will later recieve based on its calculated SES diversity.
- **District Class** A district object has a road list and a school list which collect all the road and school objects at data intake. Further, a district contains an assignment list of length equal to the number of roads, initially empty, which is the solution list described above. Lastly, it has a list of length equal to the school list, which keeps a running total of the attendance assigned at each school, in order to dynamically prevent school capacity overflow during the assignment process.
- **Assignment Class** The assignment object serves as a pseudo-wrapper class for the assignment list in the District class. It has a method to calculate its fitness based on the SES diversity of the schools in its solution, and has an instance to store that fitness value.

Fitness

In Lena's experiments, she produced solutions based on three different objective functions, each of which differently valued the relative importance of reducing commute time and maximizing school SES diversity. We used only one of her weighting functions in our experiments, $W(S) = \sum_{i=1}^{3} \left|(\frac{1}{3} - SES_i)\right| + 1$, which gave the highest relative weight to SES diversity over commute time. Here the weighting function takes a school that has already been assigned students and returns a positive value $W(S) \in [1, \frac{13}{3}]$. We define SES_1 as the proportion of students in the school with a low SES status, SES_2 as the proportion of medium SES students, and SES_3 as the proportion of high SES students. By definition then $SES_1 + SES_2 + SES_3 = 1$.

By combining the commuting distances and populations stored in our road objects with the calculated school weights, we can find the total fitness $F(A)$ of an assignment A by taking

$$F(A) = \sum_{j=1}^{S} \sum_{i=1}^{R} T_{ij} P_i W(S_j)$$

Where S is the number of schools, R is the number of roads, T_{ij} is the commute time from road i to school j, P_i is the population of road i, S_j is school j, and W is as described above.

We see here that a higher fitness actually corresponds to a worse solution, in a way that is counterintuitive to the traditional connotation of fitness. We will refer hereafter to "improving" the overall fitness as a goal of our algorithm, with the understanding that fitness improves by decreasing the value of F rather than increasing it, and that the most fit assignment in any given context is the individual with the smallest image under F.

3 Experiments

For our algorithm we developed a single crossover and a single mutation method, but we only implemented the mutate method in our experiments due to an implementation challenge in the crossover method. We defined convergence through the number of consecutive generations without improvement in the fitness of the most fit individual, and allowed the process to run without a time limit.

Genetic Mechanisms

Crossover To crossover two assignments, we simply chose a random integer in the range of 20% to 80% of the size of the assignment list, and used it as a crossover point for the two lists. That is, if n is our crossover point, we create a child using the first $n + 1$ entries of the first parent and the remaining entries from the second parent.

Using our example from earlier, which we call Parent 1, with a second example, Parent 2, we will demonstrate how we might use this crossover point to create a child.

$$\begin{aligned}
\text{Parent 1:} &\quad [4, 5, 5, 3, 1, 5, 4, 3, 3, 4] \\
\text{Parent 2:} &\quad [3, 1, 1, 4, 4, 3, 1, 5, 5, 5] \\
\text{Crossover point} &= rand(.2(10), .8(10)) = 4 \\
\text{Parent 1:} &\quad [\mathbf{4}, \mathbf{5}, \mathbf{5}, \mathbf{3}, \mathbf{1}, 5, 4, 3, 3, 4] \\
\text{Parent 2:} &\quad [3, 1, 1, 4, 4, \mathbf{3}, \mathbf{1}, \mathbf{5}, \mathbf{5}, \mathbf{5}] \\
\text{Child:} &\quad [\mathbf{4}, \mathbf{5}, \mathbf{5}, \mathbf{3}, \mathbf{1}, \mathbf{3}, \mathbf{1}, \mathbf{5}, \mathbf{5}, \mathbf{5}]
\end{aligned}$$

After the random function returned 4 from the range $\{2, 3, 4, 5, 6, 7, 8\}$, we chose all the entries up to index 4 from Parent 1, and all the entries in indices greater than 4 from Parent 2, and combined them to form the child.

[3] No pun intended

Note that while we do not require that a minimum number of children be assigned to each school other than that it is non-negative (as illustrated by our examples leaving the hypothetical school 2 empty), we do require that the number of students attending every school stay below 125% of the capacity given in the CMS data.

When applying this constraint to the crossover method, we chose to repeat the crossover with a new random crossover point if the any schools in the current assignment are more than 125% over capacity. However, because of restricted range of the crossover point, the first 20% of the first parent and the last 80% of the second parent will always appear in the child. If these portions cause an overcapacity assignment in even just one school, the crossover method will enter an infinite loop.

In present review of the project at the time of writing, it seems clear that simply removing the 20%/80% restriction might have fixed this bug, and allowing different parents to attempt to crossover if a selected pair failed a certain number of times would have certainly given the method a better chance at functioning. However, when the bug appear, we had no ability to foresee how many more challenges we would face before producing results. Given our desire to have at least a running algorithm, we proceeded with mutation-only genetic programming, and afterward chose not to return to the crossover method given the success of mutation-only reproduction.

Mutate A single mutation operation consisted of a number of element-wise swaps in the parent. Given a parent solution of list length n, we would randomly generate k between $0.001n$ and $0.002n$. Then, we would swap random elements from the entire list $\lfloor k \rfloor$ number of times. Demonstrating mutation with the actual algorithm would require a list of size at least 500, so for the sake of illustration we will change k to be between $0.1n$ and $0.2n$, and mutate our running example solution.

$$\begin{aligned}
\text{Parent} \quad & [4,5,5,3,1,5,4,3,3,4] \\
\text{\# of Mutations} \quad &= \lfloor rand(0.1, 0.2)(10) \rfloor = \lfloor .015(10) \rfloor = 1 \\
\text{Mutation 1 position 1} \quad &= \lfloor rand(0,9) \rfloor = 6 \\
\text{Mutation 1 position 2} \quad &= rand(0,9) = 1 \\
\text{Parent} \quad & [4, \mathbf{5}, 5, 3, 1, 5, \mathbf{4}, 3, 3, 4] \\
\text{Child} \quad & [4, \mathbf{4}, 5, 3, 1, 5, \mathbf{5}, 3, 3, 4]
\end{aligned}$$

If this child has an overcapacity assignment to either of the two schools involved in the swap, then the child is discarded and the parent attempts to mutate again. Though this challenge is similar to the one that caused us to abandon our crossover method, we need only check two schools for overcapacity assignments here, as opposed to having to check every school as in the crossover method. We also have no restriction on our mutation points, so for a given parent, no school will necessarily be involved in the mutation, thus avoiding the infinite loop bug in our crossover method.

Selection and Evolution

We began by generating 100 random solutions, and created successive generations of size 100 using a tournament selection algorithm outlined in Miller et al.'s "Genetic Algorithms, Tournament Selection, and the Effects of Noise." to evolove the population until it converged (Miller et al. 1995).

Tornament Selection Tournament selection chooses individuals from the population to reproduce by splitting a population of size n into some number of groups from 1 to n and then choosing the most fit individual in each of these groups as the winner of that tournament. All of these winners then reproduce – mutate, in our case – and their offspring enter the next generation.

When choosing the size of the tournaments, we encounter a conflict between inter-generational diversity and selection pressure. For large participation in each tournament, few solutions win, so there is a high selection pressure, but the diversity of the population as a whole quickly falls. For low participation in each tournament, diversity remains high, but more solutions survive to reproduce, so selection pressure falls. For example, in a population size of 100, a tournament participation size of 50 (2 tournaments with 50 participants) allows for two highly fit victors to send offspring to the next generation, but the genetic diversity of the following generations quickly shrinks. Conversely, a tournament participation size of 5 (20 tournaments, with 5 participants each) allows many individuals to survive, but only requires them to be more fit than four solutions to do so, thus decreasing the average fitness of the population, at least in the short term. In general, we see that

$$t \cdot p(t) = s(g)$$

where t is the number of tournaments, $p(t)$ is the number of participants in each tournament, and $s(g)$ is the size of the current generation. We chose to take the middle ground in this trade off, and set the tournament size to be about equal to the square root of the size of the population, so $t = \lfloor \sqrt{s(g)} \rfloor \Rightarrow p(t) = \frac{s(g)}{\lfloor \sqrt{s(g)} \rfloor} \approx \lfloor \sqrt{s(g)} \rfloor = t$. Thus $t \approx p(t)$, and we take the most moderate approach.

Evolution In our implementation, we decided to keep the generation size constant, except for the special case of the first generation. To create generation $k+1$ from generation k, we produce a combination of new individuals and mutated children. Ten percent of the time, we send a newly spawned random individual to the next generation. The other 90% percent of the time, we take the winners of the $t = \lfloor \sqrt{(s(g))} \rfloor = 10$ tournaments and randomly select one solution at a time from this group of champions. We then have the selected champion mutate, and send the mutation to generation $k+1$. We continue this process until the new generation has 100 members. Lastly, we find the most fit member of the group of tournament champions in generation k and send this "king" to generation $k+1$, thereby allowing it to survive until the next generation. Thus, the first generation has 100 members, every following generation has 101 members. The number of tournaments t is always $10 = \lfloor \sqrt{101} \rfloor = \lfloor \sqrt{100} \rfloor$, and the number of participants $p(t)$ is always $\frac{101}{10} = 10$ (by integer division).

We let successive generations run and record at each generation the fitness of the king, the fitness of the "emperor,"

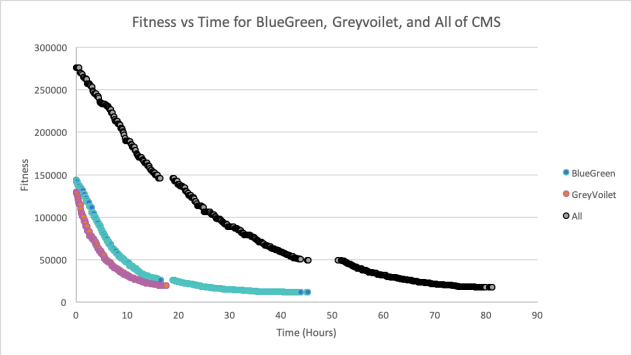

Figure 5: The graph shows 2,237 total data points representing 2,237 generations across three data sets over nearly 80 hours of runtime. The horizontal gap in the bluegreen data set and the two similiar gaps in the black "All" dataset correspond to time intervals when my computer temporarily entered sleep mode or otherwise stopped running. Thus, the actual code ran for perhaps five or six hours less than the 80+ hours the far-right black data points would indicate.

the most fit individual ever produced, along with the time it took to produce the generation, and the total running time of the algorithm. If the fitness of the emperor does not improve in 51 generations, we consider the algorithm to have converged, and we stop it.

4 Results

We ran our evolution algorithm on the three supplied files: the bluegreen and greyvoilet partitions, and the whole CMS school district. Both partitions converged, but the full district has continued to run and improve its fitness up to the time of writing.

We see that in figures 5 and 6 the combined dataset starts with about twice the starting fitness of the partitions, but improves at about the same rate. Given enough time, the combined dataset reaches comparable absolute values of fitness, and outperforms both partitions in terms of percent change from the starting value. Though the full dataset represents a larger problem, it also offers the algorithm more freedom in assigning roads to schools, especially near the boarders of the partition. For example, when the algorithm runs on the BlueGreen dataset, it might struggle to place a road on its boundary that would greatly improve the SES diversity of a school in the GreyViolet partition for only a small commute cost, but it has no ability to make this placement. In contrast, the algorithm could easily make this assignment when running on full dataset, and would likely converge to such a beneficial decision quickly. This advantage offers one explanation the larger problem's superior percent improvement.

Whatever the reason for its success, its low fitness values underscore its promise, especially given that both the partitioned datasets converged at the time of comparison, while the full data set is still evolving and improving its solutions at the time of writing.

Given more time, we would like to geographically represent our most fit final assignments across the three datasets, and compare them to geographical visualizations of solutions Lena's linear programming approach produced.

Dataset	Fit_{inital}	Fit_{final}	Size	% Diff
Bluegreen	143,567	11,429	25,274	92.0
Greyvoilet	129,450	19,305	21,456	85.1
All	275,967	17,069	46,730	93.8

Figure 6: Here $\text{Fit}_{initial}$ is the fitness of the most fit individual of generation 1, and Fit_{final} is the fitness of the most fit individual after convergence (at the time of writing for the full dataset).

To more rigorously evaluate our results, we could rerun Lena's MATLAB code and directly compare the final objective function valuations of her solutions to our converged fitness values, which are outputs of the same function. We could appraise our solutions through two additional metrics Lena describes in her thesis: the average commute time for all students across the CMS, and the number of schools who have a low-SES proportion greater than 90%.

5 Conclusions

Though we have yet to directly compare our results to Lena's, we have good reason to think ours may compete and even outperform hers. Though our algorithm has run for more than three and a half days, the CMS is one of America's largest school districts, so the potential to apply genetic programming solutions to other districts across America remains viable. The true struggle to desegregate CMS schools has political origins and contemporary political challenges, but our results can contribute to this discussion by showing that schools can be desegregated without requiring exorbitant commute times.

When the CMS reconsiders its school assignments, we hope work with Lena to leverage her contacts within the CMS to obtain real student data for our algorithms, rather using estimates from the Census and other sources. We further hope that we can present our solutions to the CMS school board, and advocate for their implementation.

6 Acknowledgements

I would like to acknowledge Lena Parker for her enthusiasm in supporting my investigation of this problem, and her willingness to answer questions regarding the data and travel to see our preliminary results at the Davidson College Verna Case Symposium May May 8th, 2019.

I would like to thank Dr. Ramanujan for his availability to answer questions, and for teaching me the methods employed here.

Lastly, I would like to thank my parents for their support through the semester and for proofreading this paper.

References

Grundy, P. 2017. *Color and Character: West Charlotte High and the American Struggle Over Educational Equality*. University of North Carolina Press.

Miller, B. L.; Miller, B. L.; Goldberg, D. E.; and Goldberg, D. E. 1995. Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems* 9:193–212.

National Center for Education Statistics, U. S. 2016. Enrollment, poverty, and federal funds for the 120 largest school districts, by enrollment size in 2016. *Digest of Education Statistics*. Retrieved on May 16, 2019.

Parker, L. 2017. A linear programming application for CMS student assignment boundaries. *Davidson College Center for Interdisciplinary Studies*.