OPTIMIZATION OF EMPLOYEE WORKING SCHEDULE AMID COVID-19

A. MASANIKA[1], A. SPIVAK[2], C. HANRATTY[3], W. M. ABDULLAH[4]

ABSTRACT. We report on a two-week COVID-19 motivated project to create employee schedules aimed at limiting the percentage of the workforce in the workplace at any time while keeping associated commute time to a minimum. The project was structured as an optimization problem. Genetic algorithms were explored, adapted and implemented. Preliminary results suggest this as a promising avenue justifying further efforts.

1. INTRODUCTION

Due to the outbreak of COVID-19 around the world, and government policies implemented as a response to the outbreak, many corporations have chosen to let their employees work from home to prevent the spread of the disease. In order to re-open the economy safely with respect to the disease threat, one of the recommendations from health authorities is to allow only a limited percentage of workers in the workplace at any specific time. Given these constraints, it is useful for companies to arrange flexible work schedules so that the employees are at the workplace during reasonable working hours. With changed schedules and perhaps shorter work shifts come the risk of increased employee commute time or increased ratio of commute time to work shift length. Thus it is desirable that these new schedules be designed to reduce commute time as well. Another objective in reducing commuting time is directly COVID-19 related: less time commuting poses a reduced risk of an employee being exposed to the disease.

In this project, we design a mathematical model to address such a scheduling problem. We implement a genetic algorithm to generate employees' working-at-office schedule for a model business satisfying certain reasonable criteria described in what follows and in the following section.

We fix a workplace, the Salesforce Tower in San Francisco with a variable number of employees. The city is divided into 39 neighborhoods numbered 0-38 and each employee's residence is identified with a number in this interval. Using Uber traffic data for n employees whose neighborhood locations are randomly generated, we seek to reduce commute time while maintaining a fixed range on work shift length and aiming to keep the maximum percentage of employees in the workplace at any time less than 35%.

We organize the rest of this report in the following way. In Section 2, we discuss the mathematical representation of the problem followed by a description of our algorithm to solve the stated problem. Results from numerical experiments on a standard collection of test instances are provided in Section 3. Conclusions drawn from these preliminary studies follow in Section 4.

2. Mathematical Model and the Genetic Algorithm

The mathematical representation of the problem and the algorithm deployed to solve the problem are discussed in this section.

2.1. **Model.** This section is devoted to providing a mathematical model of the optimization problem. We used the following parameters in our model.

- The total number of employees $n = 200 \sim 1000$
- Zone of the company S_c
- Employees'addresses $S_e^i (i \le i \le n)$
- The maximum percent of workers at the office aimed for is $\alpha = 25 \sim 35\%$
- Working time window $7AM - 9PM$
- Employee's weekly working hours $T = 15 \sim 25$ hours
- Weekdays $D = \{M, T, W, Th, F\}$
- A tuple of schedules for the employees $x = (x_1, \ldots, x_n)$

The objective function is

$$f(x) = f_1(x) \times f_2(x) \tag{2.1}$$

such that

- $f_1(x)$ is the mean daily travel time for the employees in minutes
- $f_2(x)$ is the highest percent of employees in the office at any time

The goal is to minimize $f(x)$. Any slight increase of $f_1(x)$ and/or $f_2(x)$ (i.e., closer to α) will increase the value of $f(x)$. Our goal is to minimize total commute time subject to the constraint that the workforce presence in the office at any time should not exceed 35% of the total number of employees. To this end, we chose our objective function to be $f = f_1 \times f_2$ and we examine the output produced by the algorithm for solutions that meet the desired percentage.

The following criteria were stipulated:

- Each employee must have precisely one shift per day
- No specific working sub-groups that have to be present at a given time (e.g., incorporating meeting schedule)

2.2. **The Algorithm.** To solve our problem we use a genetic algorithm. Our goal is to minimize the objective function (also known as *fitness*) $f(x)$. We have presented our algorithm (pseudocode) to solve the problem in Algorithm 2.1.

Algorithm 2.1 Our Algorithm (*DataSet*)

```
1: Initialization: Possible solutions are created from the given DataSet
2: Evaluation: Evaluate an individual using the objective function $f(x)$
3: loop <until the algorithm meets the stop criteria>
4:     Selection: Pick two individuals as parents
5:     Recombination: Apply crossover between parents
6:     Mutation: Randomly (15%) change the newly created individual (child)
7:     Evaluation: Apply Step 2 (Evaluation) to the parents and the child. If the child is
       better than any of the parents then replace that parent by the child
```

2.2.1. *Description.*

- **Initialization:** There are a number of individuals in the initial population. An individual x is a solution or a weekly work at office schedule for all employees. First we set the population size or the number of individuals required in the population. Then our algorithm starts generating individuals. For an individual, the algorithm selects a start time and a shift length for each working day for each employee (see, Figure 3). During initialization, our algorithm evaluates the value of objective function for each individual and finds the *best* individual among initial population.
- **Evaluation:** Now using Equation 2.1, we can calculate the mean daily travel time for the employees in minutes as well as the highest percent of employees in the office at any time. Therefore, we can get the value of the objective function $f(x)$ by multiplying these values.
- **Generation:** The number of iterations allowed in the algorithm is known as generation. On the other hand, we can consider this as the stop criteria. In each generation, our algorithm has the following phases: Recombination, Mutation and Evaluation.
- **Recombination:** In our algorithm, we use crossover as a recombination procedure. This is also known as one of the "mating" processes for two selected parents. First, we pick two individuals A and B randomly from the population. Then the child is produced by taking each employee's schedule from either of its parents in the following manner. Define
$$p(A,B) = \frac{f(A)}{f(A)+f(B)}$$
to be the probability of A "passing on its genes" compared to B, based on relative fitness (which parent has the lowest objective function value). Then each row in $A \times B$ should be coded to have a $p(A,B)$ chance of taking the corresponding row from A and a $1-p(A,B)$ chance of taking the row from B.
- **Mutation:** We allow a 15% chance of mutation for which one row of a matrix takes on a random row within the starting parameters. Once offspring are generated, then we keep the best two out of three of the offspring and two parents.
- **Evaluation and selection:** In this step, the algorithm compares the objective function values of the parents and the child. Let f_A, f_B, f_C denote the value of the objective functions for the parents A, B and the child C respectively. If f_C is less than f_A and/or f_B, then C replaces the parent with larger f value. Again, if C beats

either of its parent, then we compare C with the *best* solution and update the *best* accordingly. Therefore, the population converge to a better solution and the population size remain the same.

- **Output:** The algorithm returns *best* as a solution after meeting the stop criteria.

3. Numerical Testing

In this section, we provide results from numerical experiments on selected test instances.

3.1. **Data Sources.** The data set for the experiments was generated from the Uber Movement website [4]. Our original intention was to consider a city in Canada which would be familiar to the PIMS program participants. Our first assumption was that mean travel time for any trip would not vary much with the day of the work week but rather, significant differences would be found based on the time of the day of travel. We focused on the downloadable CSV files entitled Travel_Times_by_Hour_of_Day (Weekdays Only) since as the name indicates, this is precisely the data we were looking for.

Unfortunately regardless of the city chosen, there were represented only between 0-5 hours of departure for trips between any fixed origin and any fixed destination and those hours were not well chosen to capture commute times. This is reflected in our finding that it appears that for whatever city one searches for on the Uber Movement, regardless of its size and population, and whatever point of departure one chooses in that city, downloaded file, contains exactly 1,048,576 rows or trips.

Because the data was limited, we turned then to the Filtered Data on the website. The Filtered Data does not provide zones but it does include trips to a chosen address from many neighbourhoods, given by name. In particular, we chose San Francisco city in California, USA, where other data, including geographic maps, extensive census information, and information on distances between major sites, were available and could be evaluated for reliability. With the combination of our original Uber data, the neighbourhood names from the filtered data, and information from the 2016 US census, we were able to get for each of the 39 neighborhoods, average travel times for the five different times of the day: AM Peak Hours (7am-10am), Midday (10am-4pm), PM Peak Hours (4pm-7pm), Evening (7pm-12am), Early Morning (12am-7am).

We chose as our office location the single largest office building in San Francisco, the Salesforce Tower (see, Figure 1). We were able to get approximately 75% of the data points this way, i.e., average commute time to and from each of the 39 neighborhoods for each of five periods of the workday as listed in the previous paragraph. The remainder were obtained from the available data by comparison with regard to distances and identifying similar traffic conditions.

3.2. **Input and output formats of our algorithm.** In Figures 2 and 3 each row represents one of the 200 randomly chosen employees. In Figure 2, the entries in the left-hand column identify the neighborhood in which the corresponding employee lives. For example, the

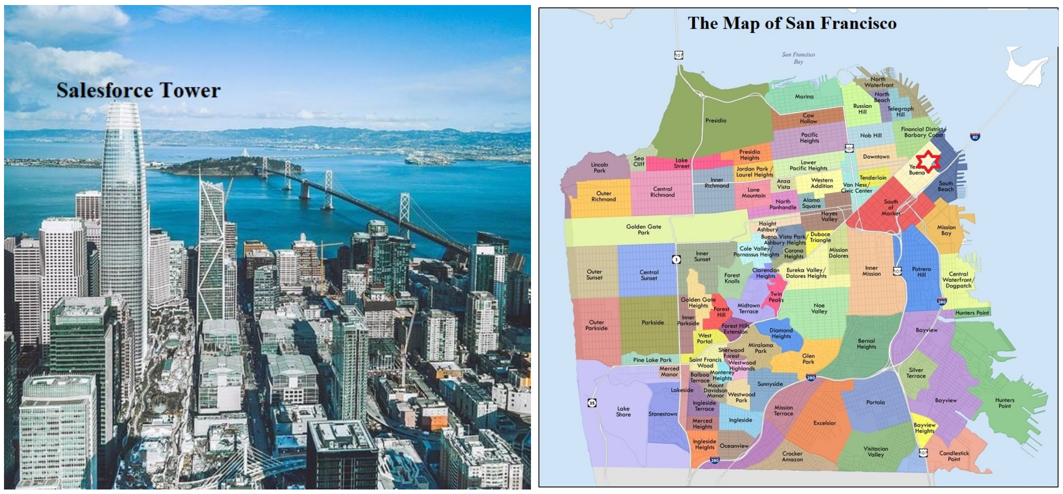
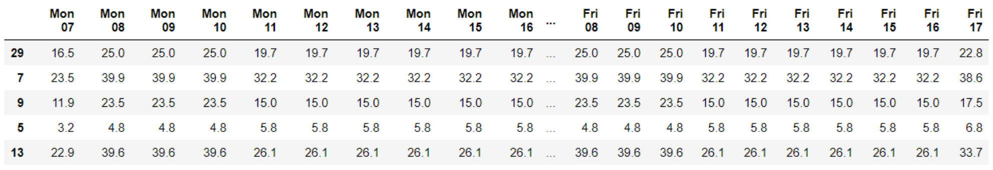


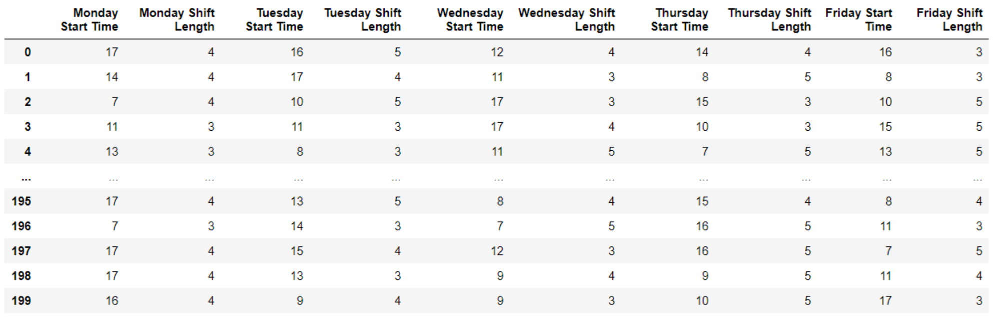
FIGURE 1. Location of the model work place and neighborhood

first employee lives in neighborhood 29, and the entries represent the commute time for each hour in a workday. Figure 3, shows the output of our algorithm indicating the start time and shift length for each employee.



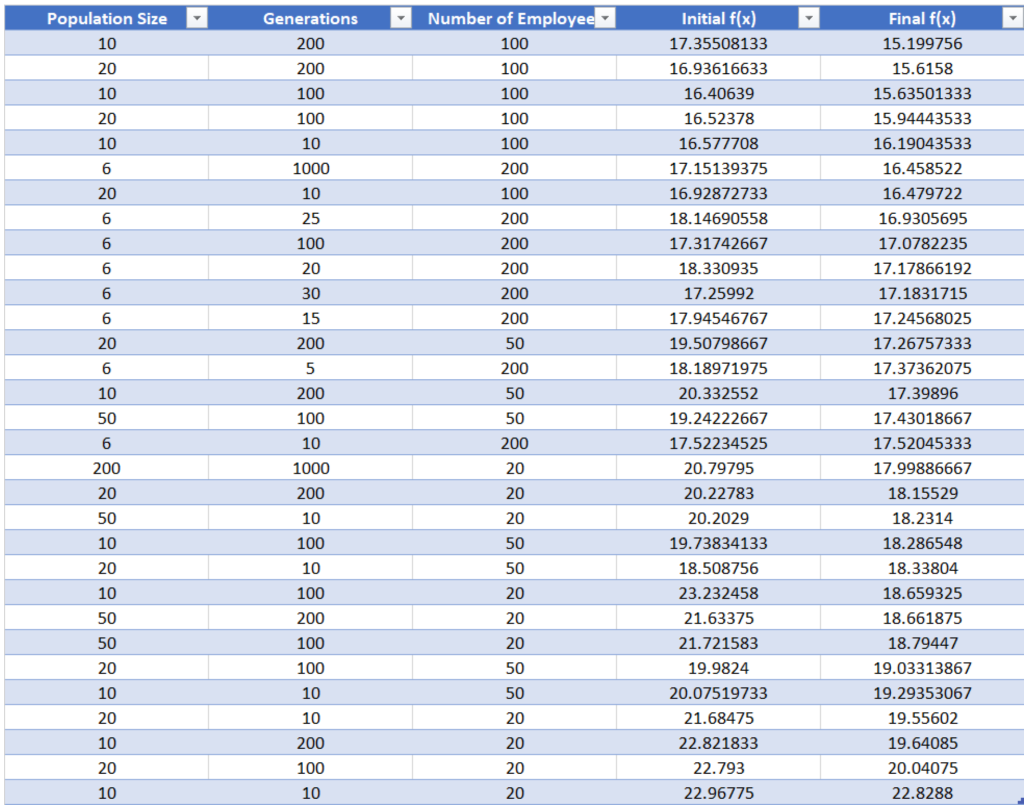
	Mon 07	Mon 08	Mon 09	Mon 10	Mon 11	Mon 12	Mon 13	Mon 14	Mon 15	Mon 16	...	Fri 08	Fri 09	Fri 10	Fri 11	Fri 12	Fri 13	Fri 14	Fri 15	Fri 16	Fri 17
29	16.5	25.0	25.0	25.0	19.7	19.7	19.7	19.7	19.7	19.7	...	25.0	25.0	25.0	19.7	19.7	19.7	19.7	19.7	19.7	22.8
7	23.5	39.9	39.9	39.9	32.2	32.2	32.2	32.2	32.2	32.2	...	39.9	39.9	39.9	32.2	32.2	32.2	32.2	32.2	32.2	38.6
9	11.9	23.5	23.5	23.5	15.0	15.0	15.0	15.0	15.0	15.0	...	23.5	23.5	23.5	15.0	15.0	15.0	15.0	15.0	15.0	17.5
5	3.2	4.8	4.8	4.8	5.8	5.8	5.8	5.8	5.8	5.8	...	4.8	4.8	4.8	5.8	5.8	5.8	5.8	5.8	5.8	6.8
13	22.9	39.6	39.6	39.6	26.1	26.1	26.1	26.1	26.1	26.1	...	39.6	39.6	39.6	26.1	26.1	26.1	26.1	26.1	26.1	33.7

FIGURE 2. The input format



	Monday Start Time	Monday Shift Length	Tuesday Start Time	Tuesday Shift Length	Wednesday Start Time	Wednesday Shift Length	Thursday Start Time	Thursday Shift Length	Friday Start Time	Friday Shift Length
0	17	4	16	5	12	4	14	4	16	3
1	14	4	17	4	11	3	8	5	8	3
2	7	4	10	5	17	3	15	3	10	5
3	11	3	11	3	17	4	10	3	15	5
4	13	3	8	3	11	5	7	5	13	5
...	...	...	...	...	...	...	...	...	...	...
195	17	4	13	5	8	4	15	4	8	4
196	7	3	14	3	7	5	16	5	11	3
197	17	4	15	4	12	3	16	5	7	5
198	17	4	13	3	9	4	9	5	11	4
199	16	4	9	4	9	3	10	5	17	3

FIGURE 3. The output format



Population Size	Generations	Number of Employee	Initial f(x)	Final f(x)
10	200	100	17.35508133	15.199756
20	200	100	16.93616633	15.6158
10	100	100	16.40639	15.63501333
20	100	100	16.52378	15.94443533
10	10	100	16.577708	16.19043533
6	1000	200	17.15139375	16.458522
20	10	100	16.92872733	16.479722
6	25	200	18.14690558	16.9305695
6	100	200	17.31742667	17.0782235
6	20	200	18.330935	17.17866192
6	30	200	17.25992	17.1831715
6	15	200	17.94546767	17.24568025
20	200	50	19.50798667	17.26757333
6	5	200	18.18971975	17.37362075
10	200	50	20.332552	17.39896
50	100	50	19.24222667	17.43018667
6	10	200	17.52234525	17.52045333
200	1000	20	20.79795	17.99886667
20	200	20	20.22783	18.15529
50	10	20	20.2029	18.2314
10	100	50	19.73834133	18.286548
20	10	50	18.508756	18.33804
10	100	20	23.232458	18.659325
50	200	20	21.63375	18.661875
50	100	20	21.721583	18.79447
20	100	50	19.9824	19.03313867
10	10	50	20.07519733	19.29353067
20	10	20	21.68475	19.55602
10	200	20	22.821833	19.64085
20	100	20	22.793	20.04075
10	10	20	22.96775	22.8288

FIGURE 4. Algorithm Results

3.3. **Experimental Environment.** The experiments were performed on PC with 3.4 GHz Intel Xeon CPU, 8 GB RAM running Windows. The implementation language is Python and the code was compiled with a Jupyter Notebook version 6.0.3 compiler. Source code available on GitHub at [1].

3.4. **Results.** After implementing the algorithm in Python, we ran a variety of tests to establish its utility. These tests are recorded in Figure 4 in order of decreasing $f(x)$. Each test was run on a choice of $P =$ population size, $G =$ number of generations, and $E =$ number of employees. We recorded the value of $f(x)$ initially, and after the chosen number of generations.

These tests demonstrated the general trend that $f(x)$ decreases with number of generations, G. Recall that a lower $f(x)$ denotes a stronger schedule. This can be observed in Figure 5 where fix the population size at 6, fix the number of employees at 200, and plot number of generations against $f(x)$.

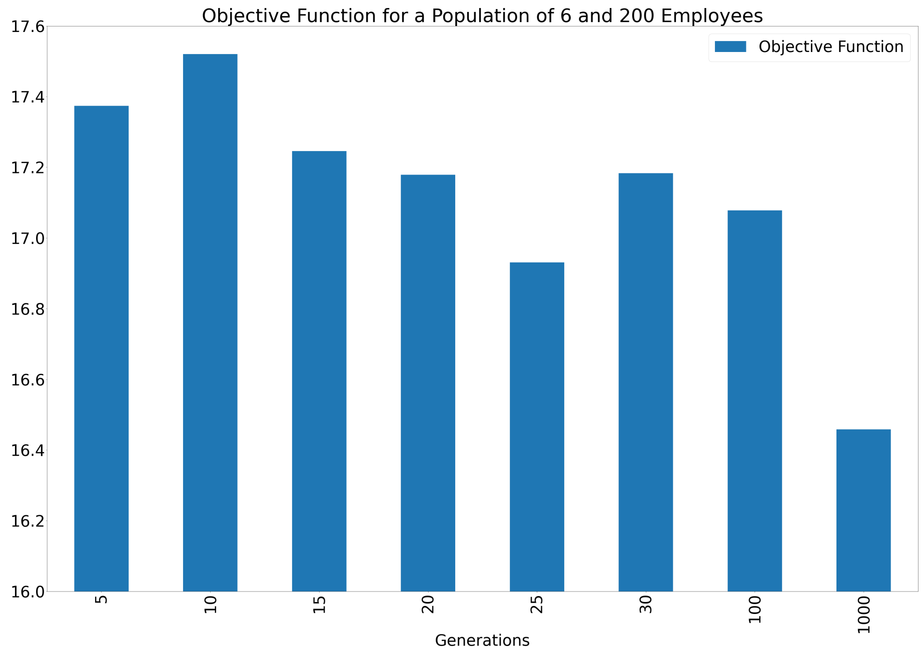
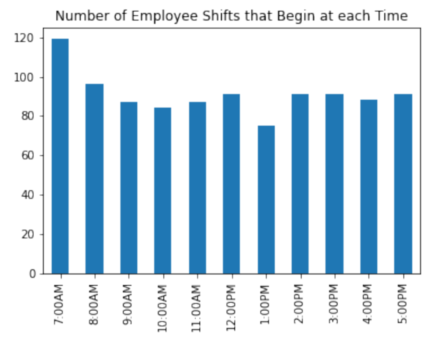


FIGURE 5. Decrease in Objective Function over Many Generations

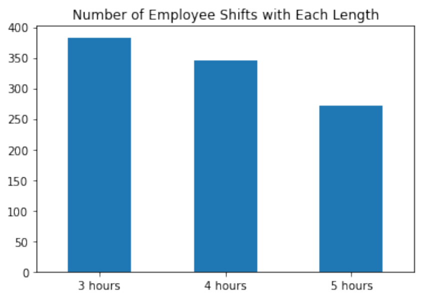
While there is a general downwards trend, we notice that some higher generation sizes also yield higher $f(x)$ (for instance, when $G = 10, 30$). This perceived discrepancy happens because each test was run on a different initial population. These fluctuations suggest that $G < 100$ is not sufficient to optimize the schedule. However, for $G = 1000$, there is a clear decrease in $f(x)$, compared to the other tested values.

In Figure 6, we take a closer look at the schedule produced after 1000 generations. Figure 6A shows the distribution of start times in this schedule. We can see that the start times are relatively balanced with the exception of the first shift in a day. We posit that this bias lowers the value of f_2. In particular, since only employees who start at 7am can be in the office at 7am, more employees are free to start at this time without negatively effecting the percent of employees in the office.

Figure 6B depicts the distribution of shift lengths in this schedule, and shows that shorter shift lengths are favoured. This will also lower the value of f_2, as shorter shift lengths means an employee will contribute to the percent of employees in the office during fewer time slots.



(A) Start Times



(B) Shift Lengths

FIGURE 6. Analysis of a 200 Employee Schedule after 1000 generations.

We conclude our analysis by considering the quality of this 1000 generations schedule. To do this, we compare this schedule to 20 randomly generated schedules, that is 20 schedules who did not undergo optimization.

In a randomly generated schedule, total commute time ranged from 43-44 minutes. After 1000 generations, the total commute time was reduced to 42.6 minutes. In a randomly generated schedule, the maximum percentage of employees in the office at any time ranged from 41 to 47 percent. Recall that the goal is $f_2 < \alpha = 35\%$. After 1000 generations, this was reduced to $f_2 = 38\%$, which is much closer to the target 35%. The test creating the 1000 generation schedule was run over night. We believe that given the time to run more generations, the algorithm could produce an even better schedule.

4. CONCLUSION

The goal of this project was to create a workforce schedule that reduces commute time, and the maximum percent of employees in the office. Such a schedule is difficult to calculate by hand for a workplace with a large number of employees. We modeled this problem to minimize the function $f(x) = f_1(x) \times f_2(x)$ where $f_1(x)$ is mean commute time and $f_2(x)$ is highest percentage of employees in the office. We deployed the genetic algorithm to find a schedule that minimizes this function. After running our algorithm for 200 employees and 1000 generations, we found an improved schedule. This shows that it is possible to automate schedule creation for large offices.

This work is an initial investigation during a two week workshop period. There are more aspects that merit investigation as described below.

Data set: As part of a longer project we would seek out other sources of data in place or to use along with the Uber Movement data.

Initial Population Generation: Currently our algorithm assigns every employee a shift every day. We would like to investigate letting employees have days without shifts, as this could decrease commute time.

The Objective Function: We ran our algorithm with one choice of objective function. We would like to try different objective functions to balance the optimization of f_1 and f_2.

The Mating Procedure: We would like to compare different mating procedures to consider larger or smaller "chromosomes". That is instead of taking entire rows from either parent, it would be interesting to compare taking individual elements from the matrix.

Time: One constraint of the genetic algorithm is its long run time. We ran a variety of tests over night, but we would like to run longer tests with higher populations and number of generations.

Algorithm: In our two week investigation we had time to consider one algorithm. This demonstrated that it is possible to use an algorithmic approach to creating these schedules. However, the genetic algorithm has a long run time so it would be valuable to compare its results to the results of other algorithms.

Acknowledgement

We thank the organizing committee of the PIMS *Math*^{Industry} Workshop 2020 for their many valuable training, guidelines and suggestions that helped to complete this project. We also thank Professors Shawn Wang and Mina Enayat who were our academic mentors. Finally, we thank Vakhtang Putkaradze, Kui Pan and Nisha Mohan from ATCO for their continuous support as our mentors from industry.

References

1. Team ATCO, *The source code of the implemented genetic algorithm*, `https://github.com/Team-Atco/COVID-Schedule-Optimization`.
2. Jason Hicken, Juan Alonso, and Charbel Farhat, *Introduction to multidisciplinary design optimization*, `http://adl.stanford.edu/aa222/Home.html`, 2012.
3. John H. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control and artificial intelligence*, MIT Press, Cambridge, MA, USA, 1992.
4. Uber Technologies Inc, *Uber movement*, `https://movement.uber.com/`.
5. Zulkifli Nopiah, Muhammad Khairir, Shahrum Abdullah, M. Baharin, and A. Arifin, *Time complexity analysis of the genetic algorithm clustering method*, Proceedings of the 9th WSEAS international conference on Signal processing, robotics and automation (2010), 171–176.

[1]University of Regina, Canada, [2]University of California, Berkeley, USA, [3]University of Alberta, Canada, [4]University of Lethbridge, Canada
E-mail address: [1]`ajm192@uregina.ca`, [2]`ameliaspivak@gmail.com`,
[3]`hanratty@ualberta.ca`, [4]`w.abdullah@uleth.ca`