Michael Encinas
Computer Engineering and Computer Science
Speed School of Engineering
University of Louisville, USA
MAENCI01@louisville.edu

***Abstract*— This project will address the hospital surgery scheduling problem by optimizing surgery schedules across multiple operating rooms while adhering to constraints such as surgeon availability, patient preparation/recovery times, and emergency surgeries. To achieve this, I will use Ant Colony Optimization (ACO) with the goal of minimizing surgery delays, reducing patient wait times, and maximizing operating room utilization. The initial implementation will focus on smaller datasets with simple constraints to evaluate the effectiveness of ACO in handling dynamic scheduling problems. Once validated, the approach will be scaled to handle larger, more complex datasets.**

***Index Terms*— Ant Colony Optimization (ACO), Hospital Job Scheduling, Meta-Heuristic Algorithms, Constraint-Based Optimization, Dynamic Scheduling.**

## I. Introduction

One of the biggest challenges facing hospitals is the efficient management of resources, especially in the context of surgery scheduling. To operate effectively, hospitals must address multiple constraints, including surgeon and staff availability, patient preparation and recovery times, and the unpredictable nature of emergency surgeries. This problem is analogous to other resource allocation challenges, such as job scheduling, supply chain logistics, and bin packing problems, where multiple tasks must be assigned to limited resources under specific constraints.

For this project, the primary objective is to minimize delays and wait times for patients while maximizing the utilization of operating rooms and staff. Traditional scheduling approaches often struggle to address the dynamic nature of real-life hospital environments, where resources may be constrained due to staffing shortages or unexpected emergencies that disrupt planned surgery schedules.

To address this, I propose using the Ant Colony Optimization (ACO) method for the scheduling algorithm. ACO is a meta-heuristic algorithm inspired by the behavior of ants in finding optimal paths. This approach is particularly well-suited for handling dynamic scheduling problems due to its ability to explore and exploit multiple solutions iteratively. By mimicking ant behavior, the algorithm can adapt to changing conditions in the hospital environment and identify near-optimal solutions that satisfy complex constraints.

The initial implementation of this project will focus on smaller datasets with simplified constraints to evaluate the effectiveness of ACO in handling surgery scheduling. The approach will then be scaled to accommodate larger datasets and more complex scenarios, mimicking real-world hospital environments. The expected results aim to demonstrate the potential of ACO in optimizing surgery schedules, contributing to improved patient outcomes and enhanced operational efficiency for hospitals.

## II. Literature review

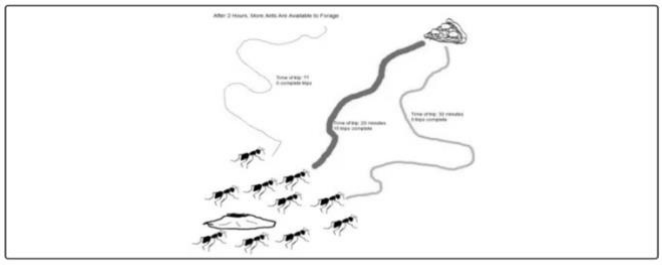
### *A. Ant Colony Optimization*

Ant Colony Optimization (ACO) is a stochastic algorithm that is designed to solve combinatorial optimization problems by mimicking the behavior of ants in finding the shortest path using pheromone trails. [1] This metaheuristic optimization algorithm has been able to handle many combinational problems, such as Traveling Salesman Problem (TSP), job shop scheduling problem, vehicle routing problem, Knapsack problem, and network routing. [2].

The general step of the ACO is to first start with initialization, which would mean placing ants on a starting point. The next step would be the movements of the ants, which would take some probabilistic function based on pheromone level. Pheromone level refers to a numeric value that simulates the behavior of real ants, leaving behind pheromone trails to communicate and guide others towards optimal paths. [3][2]

Figure 1 illustrates how ants deposit pheromones on different trails. The middle trail represents the most optimal path, determined by the shortest distance. The increased traffic on this trail reinforces its pheromone level, demonstrating its fitness as the best solution.

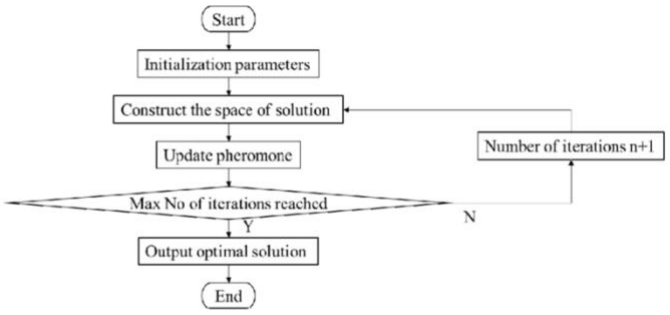
Figure 1



In step 3, as illustrated in Figure 1, the pheromone trail is updated based on the quality of the path traversed by the ant. Higher-quality paths receive more pheromone reinforcement, which can guide subsequent ants toward these solutions.

Step 4 involves the termination condition, which occurs either when a predefined number of iterations are completed, or an optimal solution is identified.

Figure 2 below, which was provided by R. Wang, displays an illustration of the process of the Ant Colony algorithm. [4]




Ant colony optimization steps.

### B. *Hospital Staff Scheduling*

Surgeons and staff scheduling is a critical challenge for the operation of a hospital. Hospitals must provide the best service to patients while being able to ensure enough coverage for surgeries. Not only do hospitals have to make sure that patients are satisfied, but they also have enough operating rooms and staff to satisfy these demands.

The complexity of this problem can be compounded by factors such as prep time and recovery time after surgery and in emergencies where there is unexpected surgery. Hospitals must balance these constraints to optimize operations and improve outcomes for both patients and staff.

According to a D. Briones article published by the American Health Law Association, hospital staffing issues are a growing concern that now hospital executives need to design, implement, and optimize up-to-date technologies to simplify and reduce redundant work to respond to staffing shortage issues. [5]

For example, when it comes to staffing issues, nurses are significantly short in the United States, according to an article in the National Library of Medicine, The US Bureau of Labor Statistics projects they will need 275,000 additional nurses from 2020 to 2030. [6]

Recent updates given by the US Bureau of Labor are provided below in Figures 3 and 4 to show that from 2023 – 2033, an additional 197,000 nurse jobs will be required to keep up with staffing, which would only require technology to be improved to be able to offset this growth if it's not met. Figure 3 below demonstrates the comparison of job growth compared to other industries.

Figure 3

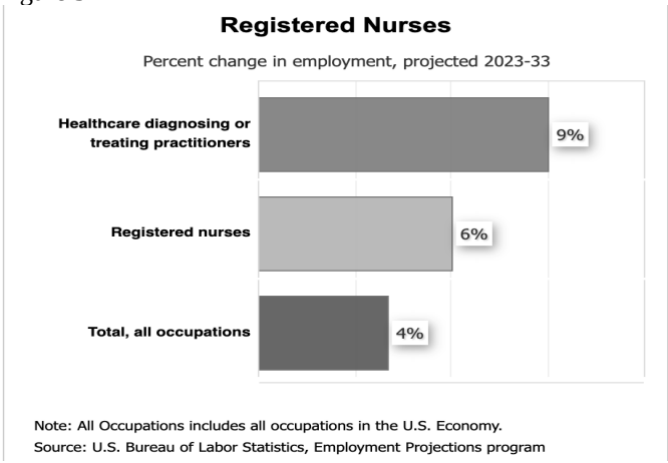



Figure 4

**Employment projections data for registered nurses, 2023-33**

| Occupational Title | SOC Code | Employment, 2023 | Projected Employment, 2033 | Change, 2023-33 Percent | Change, 2023-33 Numeric | Employment by Industry |
|---|---|---|---|---|---|---|
| Registered nurses | 29-1141 | 3,300,100 | 3,497,300 | 6 | 197,200 | Get data |

SOURCE: U.S. Bureau of Labor Statistics, Employment Projections program

Figures 3 and 4 are provided by the U.S. Bureau of Labor. [7]

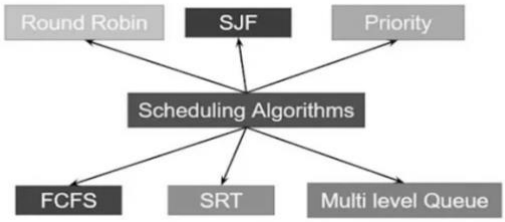
In conclusion, staffing shortages can lead to delayed surgeries, inefficient utilization of operating rooms, and reduced patient satisfaction, which can further inflate the scheduling problem.

### C. *Traditional Algorithms for job scheduling*

To better understand how Ant Colony Optimization (ACO) can provide better optimal solutions for surgery scheduling, I believe it's important to review traditional algorithms for job scheduling. The algorithms I mentioned are used in real-world scenarios such as manufacturing, logistics, and even hospital settings.

The five most popular job scheduling algorithms are First Come First Served (FCFS), Shortest Job First (SJF), Round Robin Scheduling, Priority Scheduling, and Shortest Time Remaining. [8] A visual is provided below in Figure 5. [9]

Figure 5




The reason why these algorithms could be popular is that if you are dealing with smaller datasets that are not so complex and easy to understand, then this algorithm can benefit you. If you own a small company or have a small hospital, these local search algorithms would be able to work efficiently. This is due to them being able to determine more predictable outcomes, which might be efficient in easier environments. The algorithms are also easier to implement and understand compared to more modern algorithms.

However, keep in mind as you add more constraints and adapt to challenging real-time problems, these algorithms can struggle to optimize solutions. The reason for this is that they tend to get stuck in local optima, which means they cannot obtain global optimization, which could lead to not using all resources efficiently. A common example is emergency surgeries or last-minute staff callouts.

For larger and more complex datasets, these algorithms are even less effective. As the size of the problem grows, the computational overhead increases significantly, making it difficult to efficiently manage resources and minimize delays. This limitation is of particular concern for critical high-stakes environments like hospital surgery scheduling, where the objective is to maximize resource utilization and minimize disruptions.

### *D. Modern heuristic approaches to hospital scheduling*

In recent years, heuristic and metaheuristic methods have continued to gain traction in solving complex hospital scheduling problems. The main reason for this is because traditional methods, as we discussed prior, are unable to explore large, vast search space efficiently. As well as being unable to adapt to dynamic scenarios, traditional methods are not providing optimal solutions for multi-constraint environments such as hospitals.

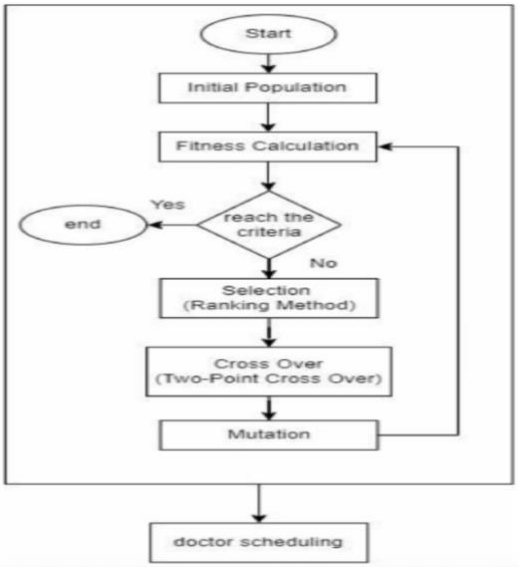
Modern heuristic algorithms can handle all the above issues that traditional methods tend to struggle with. I will be introducing some methods that have been used to help hospitals deal with scheduling, whether that be nurse scheduling, outpatient scheduling, or appointment scheduling.

For example, three modern approaches are Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Simulated Annealing (SA). These approaches in recent times have been used to help hospital scheduling problems.

Genetic Algorithm, for example, is an algorithm that is inspired by natural selection, which involves mutations, crossover, and selection. GA is renowned for its stability and efficiency in finding highly optimal solutions in very complex and large sample spaces where most other algorithms fail. [10]

A study conducted by A.Rurifando, F.Renaldi, and I.Santikarama demonstrated how they used GA to help with treating physician schedules as well as rescheduling for outpatient, inpatient, and surgeries. Here is their approach, which is shown in Figure 6.

Figure 6




Once they created an initial population, which is generated using chromosome, they then did a fitness calculation. The selection process was carried out by a ranking method. Crossovers were then introduced, which was a Two-point crossover. At which the mutation was done using a reciprocal exchange.

When their study was concluded the results, the authors stated GA was capable of effectively addressing the complex and dynamic nature of physician scheduling. They stated that optimizing parameters (ex., Pop size, crossover rate, etc), this has the potential to solve real-world scheduling approaches. The authors suggested looking into further exploration of more modern methods, such as GA with Hybrid Tabu. [11]

Just like GA, modern heuristics such as SA escape local optimum and PSO rapid convergence. The reliance on well-tuned parameters and sensitivity to problem constraints could make them less suitable for highly dynamic scheduling scenarios. This further motivates my exploration of Ant Colony Optimization (ACO), which offers a robust and adaptable framework for solving surgery scheduling problems.

## III. Proposed Approach

To achieve optimal solutions for hospital constraints and objectives, I will mimic Ant Colony Optimization (ACO). I will follow the core elements that make ACO when creating my algorithm.

For my algorithm, I will have different parameters. My parameters will consist of a different number of ants, number of iterations, pheromone decay values, alpha values, and beta values.

Pheromone decay would be my decay rate after each iteration, For example, if I set my initial pheromone levels at one and decay rate at .1, then depending on the ant's path on the next iteration, the pheromone levels would naturally go down .1, however, if certain paths are chosen then their rates should go slightly higher when the next ant is choosing its path since the pheromone scent from the last ant was left.

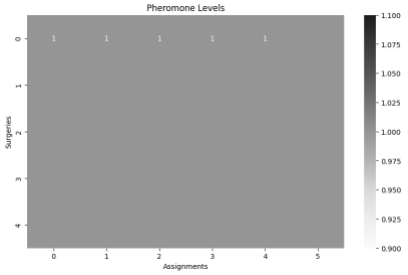
Pheromone levels (alpha), if increased, will leave more scent if a path is being chosen by an ant, which means when alpha is increased, less exploration will be made since stronger scent from pheromone trails will make ants choose paths that were stronger in scent. If alpha is low, then ants rely more on heuristic (beta) or just random exploration.

Displayed in Figure 7 and Figure 8 is an example of initial pheromone levels.

Figure 7

```
Initialized Pheromone Matrix:
     0    1    2    3    4    5
0  1.0  1.0  1.0  1.0  1.0  1.0
1  1.0  1.0  1.0  1.0  1.0  1.0
2  1.0  1.0  1.0  1.0  1.0  1.0
3  1.0  1.0  1.0  1.0  1.0  1.0
4  1.0  1.0  1.0  1.0  1.0  1.0
```

Figure 8




Once I set my decay rates and my starting influence on my alpha (pheromone), then I need to set my beta (heuristic).

Figure 9 will show you what my influence for heuristics could look like. For example, if I have a low-priority surgery that has a priority score of 2 and the duration of the surgery is 10 mins, then it would be 2/10 = .20 probabilistic chance of getting picked based on heuristic influence.

As shown in Figure 9, an illustration of how surgeries can have its heuristics probability lined up. Priority and duration are constraints that have influence.

Figure 9

```
Heuristic Values for Surgeries:
Surgery S1: 0.2333
Surgery S2: 0.0667
Surgery S3: 0.0222
Surgery S4: 0.0333
Surgery S5: 0.0467
```

Beta can control the influence of my heuristic. A higher beta can put stronger influence on heuristic influence, while lower beta can put a reliance on more scent. Beta adjusts the relative importance of heuristic constraints, such as priority weight or duration, enabling more flexible decision-making based on specific problem requirements.

For my problem, to increase exploration and exploitation, I need to have a balance to see if I can satisfy my constraints and optimize my goals, which is to be able to satisfy surgeries for that day, utilize operating room resources, and be able to handle emergency surgeries.

To find an optimal solution to my hospital surgery scheduling problem, I will analyze fitness levels, including the maximum, minimum, and mean values. These metrics will provide insight into the algorithm's performance in optimizing objectives.

## IV. Implementation

To create my algorithm, I will be using the Python language in Jupyter Notebook. I imported the following libraries: pandas, random, numpy, matplotlib, and seaborn. These libraries will help me generate my code and visualize my output.

Pseudocode

Parameter ranges

- Num_ants_value = [10,20,30]
- Num_iterations_values = [10,20,30]
- Pheromone_decay_values = [0.1, 0.3, 0.5]
- Alpha_values = [1, 3, 5]
- Beta_values [2, 4, 6]

Generate Initial Data

- Create a dataset of surgeries – Figure 10 example
  - Surgery_ID
  - Priority
  - Duration
  - Post_Time
  - Recovery_Time

Figure 10

Randomly Generated Surgeries Data:

| Surgery_ID | Duration | Priority | Post_Time | Recovery_Time |
|---|---|---|---|---|
| S1 | 21 | Low | 6 | 10 |
| S2 | 75 | Medium | 10 | 15 |
| S3 | 23 | Low | 7 | 6 |
| S4 | 83 | Medium | 13 | 12 |
| S5 | 28 | Low | 6 | 7 |
| S6 | 63 | Medium | 11 | 10 |
| S7 | 69 | High | 26 | 19 |
| S8 | 121 | High | 23 | 20 |

Create a dataset of operating rooms -Figure 11

- Room_ID
- Available_Time

Figure 11

Operating Rooms Data:

| Room_ID | Available_Time |
|---|---|
| R1 | 600 |
| R2 | 600 |

Create a dataset of surgeons – Figure 12

- Surgeon_ID
- Available_Time

Figure 12

Surgeons Data:

| Surgeon_ID | Available_Time |
|---|---|
| SG1 | 600 |
| SG2 | 600 |

Initial Matrices

Pheromone matrix set with initial values
Heuristic matrix based on surgery priority and duration (Priority weight/duration)

I set my priority levels as High (weight 7, duration 60 to 180 mins), Medium (weight 4, duration (45 to 120 mins), and Low (weight 2, duration 5 to 30 mins).

Emergency Surgeries

- Randomly create emergency surgeries
  - Surgery_ID
  - Priority set to "Emergency"
  - Duration
  - Post_Time
  - Recovery_Time

I will randomly create emergency surgeries. Weight/points for the emergency surgery is 10, with a duration between 30 to 120 minutes.

Figure 13 visualizes emergency surgeries popping up.

Figure 13

Emergency Surgeries:

| Surgery_ID | Duration | Priority | Post_Time | Recovery_Time |
|---|---|---|---|---|
| E8_0 | 114 | Emergency | 0 | 5 |
| E8_1 | 103 | Emergency | 0 | 8 |
| E8_2 | 59 | Emergency | 0 | 8 |

Probability calculation function

- Pheromones and heuristics play a big role in an ant choosing its next path.

$$CV = (PRow)^{\alpha} * (HRow)^{\beta}$$

$$P = \frac{CV}{\Sigma TotalCV}$$

PRow reflects the train strength left by previous ants (Pheromones)
HRow reflects real-time factors such as priority or duration (Heuristics)
Alpha parameters control the influence of pheromones
Beta parameters control the influence of heuristics

Fitness Evaluation

- Calculate total fitness based on
  Priority score of surgeries scheduled
  - High – 7 points
  - Medium – 4 points
  - Low – 2 points

  Bonus for emergency surgeries scheduled
  - Emergency – 10 points
- Penalties
  - Unscheduled emergency surgeries (4-point penalty)
  - Unscheduled regular surgeries
  - High – 5 points
  - Medium – 2 points
  - Low – 1 point

Fitness/Cost Function
S: Set of scheduled surgeries.
U: Set of unscheduled surgeries.
$p_i$ : Priority score of surgery $i$, where $i \in S \cup U$ (e.g., Emergency 10, High 7, Medium 4, Low 2).
$w_i$ : Penalty for not scheduling surgery $i$, where $i \in U$ (e.g., Emergency 4, High 5, Medium 2, Low 1).

$$Fitness = \sum_{i \in S} pi - \sum_{i \in U} wi$$

Goal is to Max Fitness

Main ACO – if just one parameter
For each iteration(num_iterations_value)

- Generate emergency surgeries
- For each ant(num_ants)
  - Build a schedule using pheromones and heuristics
  - evaluate the fitness of its path
  - track the best path if fitness improves
- Then, update pheromones based on schedules and fitness values.

Output – best scheduled with fitness score and what was not scheduled with best scheduled. Shows my unscheduled surgeons and what I need to fulfill that surgery. Either more rooms or more surgeons. Figure 14 shows an example of an output to visualize what I am expecting.

Figure 14

```
Detailed Schedule with Times and Priorities (Including Emergencies):
   Surgery_ID Priority  Duration  Post_Time  Recovery_Time Assigned_Room  \
0        E8_0      NaN       NaN        NaN            NaN           R1
1        E8_1      NaN       NaN        NaN            NaN           R1
2        E8_2      NaN       NaN        NaN            NaN           R1
3          S1      Low      21.0        6.0           10.0           R4
4          S2   Medium      75.0       10.0           15.0           R7
5          S3      Low      23.0        7.0            6.0           R4
6          S4   Medium      83.0       13.0           12.0           R8
7          S5      Low      28.0        6.0            7.0           R2
8          S6   Medium      63.0       11.0           10.0           R7
9          S7     High      69.0       26.0           19.0           R8
10         S8     High     121.0       23.0           20.0           R8
11         S9   Medium      97.0       11.0           19.0           R5
12        S10      Low       7.0        6.0            7.0           R3
13        S11      Low      24.0        9.0            6.0           R1
Best Unscheduled Due to Room Time: ['S40']
Best Unscheduled Due to Surgeon Time: []

Unscheduled Surgeries (Not Completed Due to Time Constraints):
Surgery_ID  Duration Priority  Post_Time  Recovery_Time
       S40        96   Medium         18             18


Best Fitness (Including Penalties): 192
Fitness Percentage: 96.97%
```

### *A. Data*

To evaluate the performance of my ACO algorithm, I created a dataset that contained 40 surgeries, 8 surgery rooms, and 12 surgeons. I decided to stimulate a medium one-day hospital schedule where there are scheduled surgeries and then random 2 or 3 emergency surgeries per day. I ensured that every iteration included at least one emergency surgery to avoid biased data that could favor iterations or ants without emergency cases, which typically score higher due to fewer constraints.

To get the parameters and constraints for my data, I read multiple articles which were focused on solving the optimalization problem of nursing staffing or hospital surgery/operating room scheduling. The parameters and constraints my data was based on were N number of surgeries which are scheduled in one operating room per day, surgeries on mean duration for scheduling, surgery duration, limited number of surgeons, maximum workload for surgeon, pre-surgery readiness, limited number of surgeons availability, nurse staffing, time periods, probability of a scenario occurring with constraints such as optimizing surgeon/nurse scheduling while handling time constraints. [11][12][13]

Data from the articles I read were based on an Out-Patient Department in a hospital in Ghana, an A-A-A general hospital with 45 departments that was collected from an orthopedics department and an hospital floor at University of Wisconsin Hospital and Clinics.

The data was randomly generated, but I made sure to make the randomly generated data be the same for every parameter run so I could see how my fitness scores behave with each unique combination.

I did 5 runs on each parameter and averaged out the following metrics: average minimum fitness, average mean fitness, and average max fitness.

A CSV file was created, ant_colony_results.csv. I will be displaying those results with visual graphs and charts.

## V. Results and Analysis

When running my algorithm, I created 243 unique parameter combinations that were placed on a CSV file named ant_colony_results.csv. I will be displaying different charts of every parameter with its unique combination that contains the metrics of average fitness, average max, and average minimum. I will look to see if I can find insights and analyze how I can obtain more optimal solutions.

Figure 15 is hard to see, but for visual purposes, you can observe all the unique combinations bunched together.

Figure 15

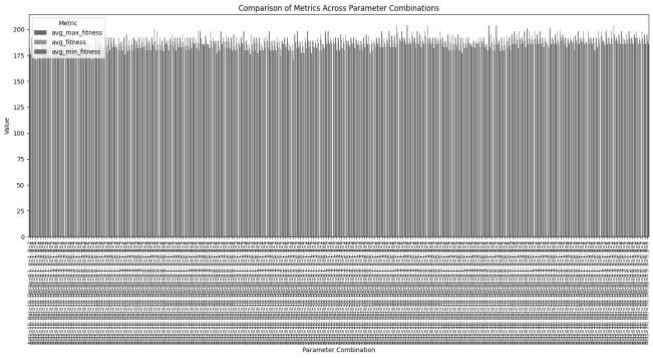
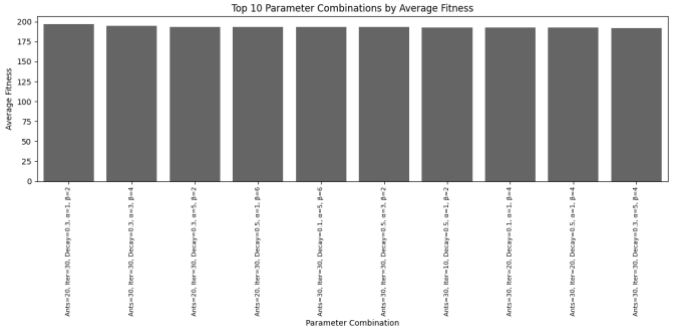
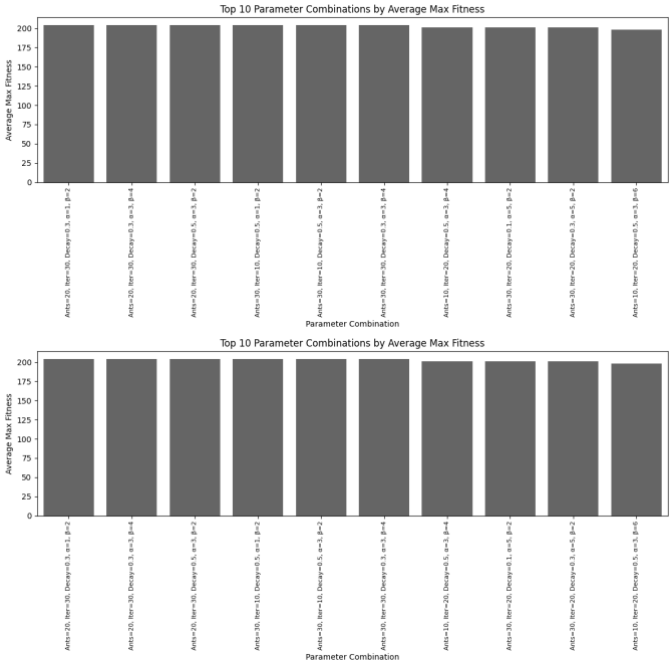



Figure 16 I made bar charts of the top ten parameter configurations for each unique metric of fitness.

Figure 16





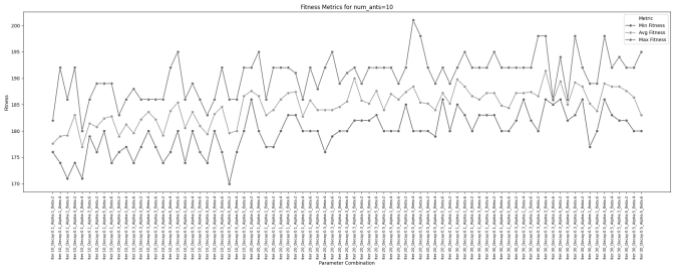

In Figure 16, when observing average fitness best results, I noticed that when the number of ants is 20 and 30 as well for when iterations are usually 20 and 30. It appears that more ants and more iterations improve fitness results. The decay rate also tends to be .5 or .3, which means lower decay (.1) does not tend to get better average fitness results.

For average max fitness, the number of ants and iterations play a big role since 20 and 30 both makeup most of the top ten parameters for the Figure 16 chart. The decay rate is still around .3 and .5, which might explain that since there is more decay than, there could be more pheromone evaporation, which will encourage exploration and prevent early convergence to suboptimal solutions. I observe that for alpha and beta, I have alpha values of 1 and 3 and beta values of 2 and 4 more often than alpha at 5 and beta at 6. This demonstrates that alpha and beta with moderate pheromone and moderate heuristics tend to give me more optimal solutions then parameters with high values of alpha or beta.

For average minimum fitness in Figure 16, the parameters appear to be the same in comparison to the top parameters for average fitness and average minimum. Looking forward, I should see trends where fitness scores go up if they fit similar parameters as the top parameters in Figure 16.

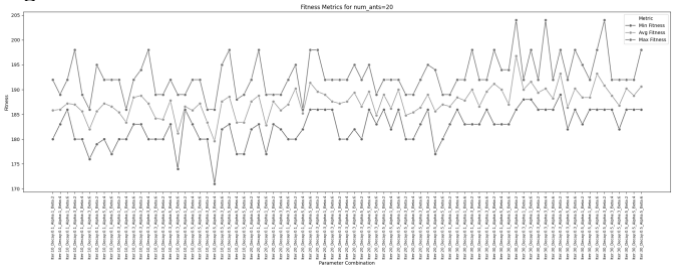
I made charts for several ants = [10, 20, 30] with each of its unique parameter configurations while displaying their metrics of average fitness, average max, and min.

Figure 17




For Figure 17, as iterations are increased, then all metrics tend to increase. For min fitness, there are huge drops between different parameters, which might mean fewer ants mean limited exploration. For average fitness, my values are more stable compared to min fitness. I do notice that decay rates of 0.3 and 0.5 appear to support better exploration and exploitation. For max fitness, the highest peaks tend to show when alpha is 3 and beta is 4. This supports the previous charts in Figure 16, with alpha and beta being moderate.

Figure 18

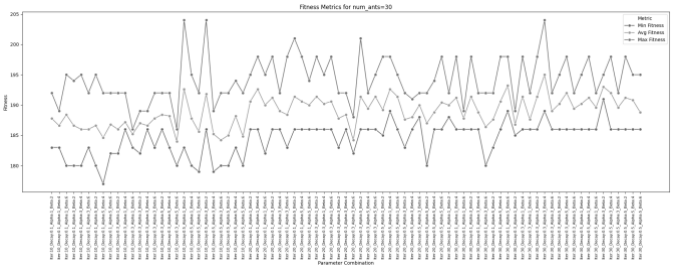



When observing Figure 18 for number of ants = 20, for the minimum fitness, the fitness values are higher than number of ants = 10, I also observed slightly fewer dips, which might suggest more ants could improve consistency and minimize poor solutions.

For average fitness, the fitness rates tend to be higher compared to Figure 18 results. I also observed that decay rates of 0.3 and 0.5 seem to correlate with better average fitness.

On max fitness results, the peaks stand out more often with moderate alphas, and betas continued to be able to have optimal solutions.

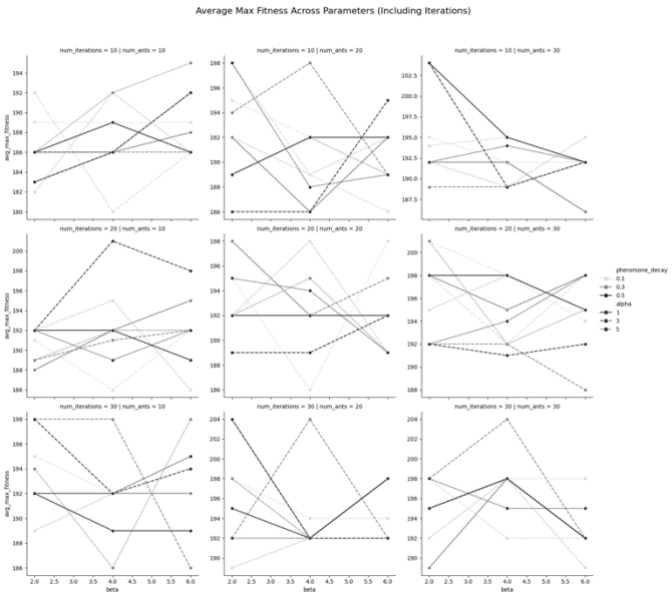
Figure 19




For Figure 19, I have the highest fitness values for all fitness metrics for number of ants = 30. The number of ants makes big improvements in having more stable and optimal solutions as well as having increased iterations. The decay rate of 0.3 and 0.5 continue to improve fitness scores.

After looking at all my charts above, I will dig deeper into very specific parameters to verify if Alpha of 1 or 3 and beta of 2 or 4 continue to have positive results. As well as decay rates of 0.3 and 0.5 having an impact, I will focus on maximum fitness for all unique parameters to see if I can find what parameters have the biggest impact on my ACO approach. Figure 20 will visualize my max fitness with all parameters.

Figure 20




When focusing on max fitness to see what parameters give me the best optimal solution, figure 20 demonstrates that the number of ants being 20 and 30 gives me the best results, as well as iterations being 30. Beta for the highest fitness charts are at 4, with an alpha of 3. Decay Rate is also set at .5.

For reference, when looking at all my metric fitness scores, this algorithm was able to schedule at least 89% of surgeries, which includes emergency surgeries. At the highest peak, I got 95% of surgeries scheduled. High fitness scores do not always mean I will have the high percentage of surgeries scheduled, but I was able to get more priority surgeries scheduled.

## VI. Conclusion

For this project, I was able to successfully implement an Ant Colony Optimization algorithm to tackle the hospital surgery scheduling problem. The algorithm demonstrated a high level of effectiveness in minimizing surgery delays, maximizing operating room utilization as well as handle emergency surgeries. Additionally, I was able to generate outputs explaining why certain surgeries were not scheduled and what resources I need to fulfill those surgeries in the future, such as more rooms or more surgeons.

I noticed that I did have time left over for operating rooms and surgeons, but I believe it's because my ants in my algorithm are choosing paths based on pheromones and heuristics more than filling all rooms if there is available room time. If I were to adopt a greedy approach with my ACO, I am sure I could get higher fitness scores depending on the data I was given, but then it would deviate away from pure ACO.

Although my dataset was randomly generated, I believe future work should focus on using real hospital data. This would provide a more accurate representation of hospital scheduling challenges, as my data only mimics real scenarios. However, I am confident that my algorithm can handle real hospital data, as it was designed with flexibility and adaptability in mind.

For my parameters, when using ACO, it would be better to use higher number of ants and iterations, as this tends to increase all fitness metrics. I also would stay with more moderate values of heuristic and pheromone as this tends to increase exploitation and exploration for optimal solutions. Decay rates also play a factor, as decay rates that lower tend to have less optimal solution results compared to higher decay rates.

Finally, while my computer's computational limits restricted the number of parameters I could test, further research could explore the impact of adding more parameters to improve optimal solutions. I averaged 5 runs for each unique parameter combination, but increasing this to 10 or more could yield more reliable results. Unfortunately, my hardware limitations prevented me from testing beyond this level.

One of the most challenging aspects of implementing my ACO algorithm for this project was determining the heuristic weights and penalties for the constraints. If too much emphasis was placed on surgery priority points for higher-priority surgeries, the algorithm might overly focus on scheduling high-priority cases while neglecting lower-priority surgeries. This could negatively impact hospital efficiency and customer satisfaction. Similarly, if the duration of specific surgeries was weighted too heavily—whether long or short—it could skew the heuristic search and result in low-priority surgeries being scheduled over high-priority ones. Penalties also played a significant role. If penalties for not scheduling certain surgeries were set too high, the algorithm might prioritize those surgeries to avoid penalties, even if it meant achieving a lower overall fitness percentage for fulfilled surgeries. Striking the right balance among these metrics was crucial to prevent bias toward any single constraint while ensuring a more efficient scheduling solution.

## References

[1] C. Ratanavilisagul and B. Pasaya, "Modified Ant Colony Optimization with Updating Pheromone by Leader and Re-Initialization Pheromone for Travelling Salesman Problem," 2018: IEEE, pp. 1-4, doi: 10.1109/iceast.2018.8434500. [Online]. Available: https://dx.doi.org/10.1109/iceast.2018.8434500

[2] Induraj, "Implementing Ant Colony optimization in python-solving Traveling Salesman Problem," *Medium*, 2023. [Online]. Available: https://induraj2020.medium.com/implementation-of-ant-colony-optimization-using-python-solve-traveling-salesman-problem-9c14d3114475.

[3] Y. Pei, "2012 International Conference on Computer Science and Electronics Engineering," *2012 International Conference on Computer Science and Electronics Engineering*, 2012, doi: info:doi/10.1109/ICCSEE.2012.178.

[4] R. Wang, "Research on Robot Path Planning Based on Improved Ant Colony Algorithm under Computer Background," *Journal of Physics: Conference Series*, vol. 1992, p. 032050, 08/01 2021, doi: 10.1088/1742-6596/1992/3/032050.

[5] D. A. Briones, "Staffing Shortages: Responses and Risks at Hospitals and Health Systems." [Online]. Available: https://www.americanhealthlaw.org/content-library/publications/briefings/d4102f32-1be8-45bb-9cfe-a35cc59763f7/staffing-shortages-responses-and-risks-at-hospital

[6] P. A. Lisa M. Haddad, Tammy J. Toney-Butler, "Nursing Shortage," *National Library of Medicine*, 2023. [Online]. Available: https://www.ncbi.nlm.nih.gov/books/NBK493175/.

[7] United States of America. (2024). *Registered Nurses*. [Online] Available: https://www.bls.gov/ooh/healthcare/registered-nurses.htm#tab-6

[8] E. Staff, "A Guide to Job Scheduling Algorithms: Efficiently Managing Your Workflows," A. Batch, Ed., ed, 2023.

[9] V. Sonwalkar, "Scheduling Algorithms in Operating System." [Online]. Available: https://vsonwalkar3.medium.com/scheduling-algorithms-in-operating-system-bade5f192ca4

[10] R. Bai, E. Burke, G. Kendall, J. Li, and B. McCollum, "A Hybrid Evolutionary Approach to the Nurse Rostering Problem," *Evolutionary Computation, IEEE Transactions on*, vol. 14, pp. 580-590, 09/01 2010, doi: 10.1109/TEVC.2009.2033583.

[11] Z. Zeng, X. Xie, J. Li, H. Menakerand S. G. Sanford-Ring, “An analytical model for performance evaluation of operating room schedules in orthopedic surgery”, 2014. doi: 10.1109/coase.2014.6899383.

[12] Jun Hu, Qiang Su, Qian Wangand Qiugen, “A surgery scheduling model based on surgery grading management system”, 2017. doi: 10.1109/icsssm.2017.7996265.

[13] J. Sangai and A. Bellabdaoui, “Workload balancing in nurse scheduling problem models and discussion”, 2017. doi: 10.1109/logistiqua.2017.7962878.