Advanced Algorithms Final Project

For the last two weeks of the course, you will have an opportunity to work a project of your own choice.

The main goals of the final project are to

- allow students to choose a topic of personal interest for in-depth study and research
- allow students to further explore topics introduced in class
- provide experience in reading technical literature
- provide practice at presenting technical information (orally/visually)

A successful project will

- build upon your advanced algorithms foundation (for example, by going deeper into an AA topic already covered in this course, by investigating a new area of AA that builds upon knowledge gained in the course, or by improving key skills developed in - or relevant to - this course)
- involve something challenging, interesting, and creative/original
- involve a theoretical and implementation component

Teaming

Our hope is that you will work on teams of 2 to 3 people. However we understand that these uncertain times have created added challenges for teaming, so if you would like to work alone we will allow it.

Project Proposal (Due April 26 @ 12pm EST)

You have two options for this assignment.

If you decide to explore one of the project options that we suggest, then submit a PDF with the following information

- Who is on your team
- Which one of the option you decide to pursue
- Project plan/ timeline in terms of what you will get done for the next two weeks
- List of questions for us (if applicable)

If you decide to pursue a different project, then submit a PDF with the following information

- Project title
- Who is on your team
- Short project description
- List of at least 3 resources that you will investigate
- A breakdown of what you will accomplish in order for us to evaluate your project as successful
- Project plan/ timeline in terms of what you will get done for the next two weeks
- List of questions for us (if applicable)

Final Deliverables

Your final deliverables will include a (1) 15-minute final presentation intended to teach your peers about your project, (2) complementary report/documentation, and (3) a GitHub repository containing the implementation part of your project.

What happens on Final Day (May 5 @ 12pm EST)

This class will not have an exam period. The final event will take place during the last class period on 5/5. On the final day, each project team will give a **15-minute presentation + 3-minute Q&A** to showcase their two week work of the project . Think of it as an opportunity to teach your peers a new topic you investigated. After each team gets a chance to present, we will open breakout rooms for each team to continue Q&A or work on finishing the final report.

Final Presentation (Due May 5 @ 12pm EST)

- Each team will deliver a 15 minutes presentation intending to teach the rest of the class the necessary concepts to understand their project. Though the presentation is meant for teaching purposes, it should also include an overview of the project, what was accomplished, what obstacles were faced, and lessons learned.
- Successful presentations generally
    - Include many examples
    - Omit highly technical/complicated details in favor of providing a rich overview of new material in a digestible way; the highly technical details can be explained in your supplemental documentation
    - Use a structure that is easy to follow (typically including an overview/introduction, helpful examples, a discussion of your work, and conclusions)
    - Give credit where credit is due (for figures, graphs, theorems, algorithms, etc.)

    - Indicate how the topic builds upon your advanced algorithms foundation, what was challenging, and what was creative.

Complementary Documentation + GitHub Repo (Due May 11 @ 10am EST)

- Over the course of the project, you'll probably learn a lot of cool things relating to your topic (proofs, algorithms, etc..). The goal of the final presentation is not to go over every single concept that builds up your topic. This documentation is your chance to capture the more in depth things you learned.
- The documentation must include
    - A general introduction to your topic (a paragraph or two at the top that goes over what you'll be discussing)
    - A bibliography
    - Pseudocode and description of your implementation
    - Link to your GitHub repository with well commented code and a README with what your repo contains and how to run it
- The documentation should also include at least one of the following
    - Detailed proofs that you wrote or learned about and were key to your understanding
    - Details of certain examples that you worked through or generated
    - Any other work that was critical to the overall project
- The documentation should have a clear narrative thread (imagine that someone with just an Advanced Algorithms background is reading this, try to facilitate their learning as much as you can by not jumping around)
- (For those that took Linearity I & II and/or Discrete Math, the presentations and documentation generated for those final projects are what we're looking for)

Timeline

- Project proposals are due April 26th @ 12pm EST
- Project in class check in and work time: April 26th, April 28th, May 3rd
- Final presentations are due May 5th @ 12pm EST
- Complementary documentation + GH repo are due May 11th @ 10am EST

Potential Project Topics

Here are some project ideas that we have for you.

Idea 1: From Local Search to Social Optimum

One project idea is to expand the local search algorithm to a system with a large number of agents with their own goals and objectives. Instead of trying to find a good local optima

in a solution space, you want to find the **social optimum**, which minimizes the total cost to all agents. The tricky thing about this multi-agent system is that agents can share costs, so they aren't independent from each other. This project would involve learning about **best-response dynamics** (each agent updates based on its best response to the current situation) and the **Nash equilibrium** (stable solution to the problem, where the best response for each agent is to stay put). This project would be a cool choice if you are interested in game theory.

List of resources

- Section 12.7 Best-Response Dynamics and Nash Equilibria within *Algorithm Design* (pages 690 to 700): https://homepages.cwi.nl/~apt/stra/sharing.pdf
    - This is a good intro into the concepts mentioned in the project description
- CS364A: Algorithmic Game Theory Lecture #16: Best-Response Dynamics: http://timroughgarden.org/f13/l/l16.pdf
    - More focused on the theory side of best-response dynamics
- "Game Theory concepts with application in Python using Nashpy": https://towardsdatascience.com/game-theory-in-python-with-nashpy-cb5dceab262c
- Documentation on Nashpy Python library: https://nashpy.readthedocs.io/en/stable/tutorial/index.html
- "The Nash equilibrium: A perspective": https://www.pnas.org/content/101/12/3999
    - Provides more details on the Nash equilibrium

Things we would expect you to accomplish to have a successful project:

- General introduction to game theory, best-response dynamics, and Nash equilibria (you would probably focus on this during your final presentation)
    - At least one toy example used to explain these concepts
    - Another more complicated example that highlights a real-world use case
- Annotated bibliography containing at least 5 sources
- Simple implementation of best-response dynamics/Nash equilibria either using Nashpy or implementing a toy example from scratch
    - If you use NashPy, trying implementing something a bit more complicated
- Some nice to haves include proofs on convergence for best-response dynamic problems

Idea 2: Social Network Analysis

Start learning about some basic algorithms used in social network analysis. There are a ton of algorithms that you could explore here, including the Girvan-Newman method

(look for edges that form part of the shortest paths between pairs of nodes in different parts of a network), graph partitioning algorithms, and different ways of calculating centrality measures (e.g. betweenness and PageRank).

List of resources

- Relating to graph partitioning algorithms
    - UCSD "Four graph partitioning algorithms" lecture slides: http://www.math.ucsd.edu/~fan/talks/mlg.pdf
    - UC Berkeley notes on graph partitioning: https://patterns.eecs.berkeley.edu/?page_id=571
    - Stanford "Graph Partitioning Algorithms" lecture slides: http://adl.stanford.edu/cme342/Lecture_Notes_files/lecture7-14.pdf
- Stanford Large Network Dataset Collection: http://snap.stanford.edu/data/index.html
    - I highly recommend this source when trying to find a dataset for the implementation part of the project

Things we would expect you to accomplish to have a successful project:

- Pick 1-2 social network analysis algorithms to learn about
- Introduce each algorithm, explaining how it works and why it's useful in the real-world (you would probably focus on this during your final presentation)
    - At least one toy example used to explain these concepts
    - Another more complicated example that highlights a real-world use case
    - Explain the mechanics/ math behind why those algorithms work
- Annotated bibliography containing at least 5 sources
- Implementations for the algorithm(s)
    - If you pick 1 algorithm, try to implement at least two different scenarios and compare the two
    - If you pick 2 algorithms, try to implement both and compare the two

Idea 3: Mixed Integer Linear Programming

One possible topic is the full generalization of the original linear programming framework, the fancier *mixed integer-linear programming* (MILP), also known as *mixed integer programming* (MIP). Here, while the constraints and objective function are still linear equations, the variables themselves can be real numbers, integers, or simply binary variables. This is a very robust framework for addressing problems, **particularly problems around optimization**. Furthermore, you can think of it as "just" the combination of the linear programming and integer programming techniques you already know: 1. Use the LP relaxation to simplify any binary/integer constraints, 2. Solve the relaxation, and

then 3. Use Branch & bound or Cutting planes to address the integer aspects of the problem! We've covered all of that in the units so far, so you're very well prepared to tackle MILP problems. So where's the difficulty? idk

Well, to start, *MILP problems are NP complete*, meaning that there's no algorithm that's guaranteed to be good. So, one potential direction is to *look into Branch and Cut algorithms*, ways to solve the MILP problem ever so slightly better. But people have spent quite some time on that, so the other, and dare we say more interesting, part of MILP is **modeling a given scenario as a MILP problem**. Because MILP problems are inherently optimization problems, they work well in a variety of situations, such as path planning and controls for robots, capital budgeting for investments, scheduling (including the TSP!), etc. The challenge is to then model the problem using linear constraints and objectives for use in an MILP setting.

tl;dr: model an interesting problem as an MILP problem and then code up a solution.

List of resources

- Wikipedia: It's dense, but you'd be surprised how much is packed in here.
- Python-MIP, Gurobi, FrontlineSolver: These are all packages that use well-designed Branch-and-cut algorithms to solve MILP problems. Python-MIP is the only one of the three that's open source, but it's quite a powerhouse.
- Here are several papers on applications of MILP: General multi-vehicle path planning, Autonomous Underwater Vehicle path planning, Conflict resolution for air traffic control, Robustness evaluation of neural networks
- A good theoretical cover of MILP: it can get really theoretical, but it covers scheduling, budgeting, and warehouse location problems.
- Introduction to MIP in the context of path planning: the slides are hard to parse, but it goes along well with previous resources
- Introduction to MIP through the applications to biology and medicine: it's great! Especially if you like medicine.

Things we would expect you to accomplish to have a successful project:

- Introduce the problem you've picked:
    - Its relevance to the real world (ex. if you're doing a scheduling problem, pick a 'useful' example, not just "I have events to stick in a calendar").
    - Explain the relevant 'dynamics' of the system, and how you'll model it as an MILP system. You must include a description of relevant variables and their domains, constraints, and an objective function. Try to at least loosely justify this model with basic sanity checks.

    - Your own visualization of the 'space of feasible solutions'
- Annotated bibliography with at least 3 sources
- Implementation of the solution
    - Include a discussion on if/why your answer looks feasible. It's fine if it doesn't - it's more important to get a good model than a good implementation!

Idea 4: Vehicle Routing Problem

The Vehicle Routing Problem! A classic that was once part of the homework of last year's AA iteration, it is now upgraded to be a potential final project topic. The VRP, as it is known, is considered the generalization of the Traveling Salesman Problem, and is *NP Hard* - beyond NP Complete! It's not a trivial problem, but it has many direct applications to industry, and is consequently of great interest. While solving the VRP itself is difficult, we can still use a heuristic to find a good solution - one way being using local search algorithms. There are furthermore several variants of the VRP problem, all interesting on their own. One of those variants is the Constrained VRP (CVRP), and can be solved using the Integer Programming tools we've covered already. Regardless of what portion of the VRP problem you wish to cover, there's sufficient space in the area to get your hands wet.

List of resources

- Wikipedia: You'd be surprised how good the description of the VRP is. It details the mathematical formulation of the VRP, as well as briefly covering the variants and how they are related to each other.
- Notes from MIT on Clarke and Wright's algorithm for solving the VRP: it's dense with math, but it's a solid theoretical description and approach to solving the VRP. It also has some decent visualizations as well.
- Youtube video on using the Gurobi package in Python to solve the CVRP
- Intuitive description of the VRP and the various APIs that can be used to solve it: doesn't go into detail about either the math behind the VRP or how to code using the APIs, but it's a good place to start.
- Google's scaffolding on using their tools to solve the VRP: it doesn't really cover the details of the VRP or the math, but it helps for writing code.
- Code scaffolding from AA last year: this is scaffolding *only* for the integer programming approach to the CVRP variant of the VRP.
- A really fancy way of solving the VRP efficiently: if you're interested!

Things we would expect you to accomplish to have a successful project:

- Introduce the problem you've picked:

    - A description of the VRP in general. Ground its relevance with some real-world context.
    - A description of the specific VRP you aim to solve in the code, as well as its accompanying mathematical formulation.
- Annotated bibliography with 5 sources
- Implementation of the solution
    - If you chose to do an exact solution (using IP, etc.), add a discussion on the runtime of the algorithm, and how you'd rate its usefulness as the problem input scales.
    - If you chose to do a heuristic approach (using local search, etc.), add a discussion on how close the solution is to optimal, and when the heuristic approach fails/succeeds. Include some justification for why the heuristic chosen is good, through sanity checks and the like, and if possible, include some math :)
    - If your code is developed from the scaffolding provided, please note this. (There's no penalty if you do - that's why we gave it!)

Idea 5: Fair Carpool Sharing with Network Flows

In the Network Flows unit and lab 0, we looked at the city evacuation problem and the badminton elimination problem that can be solved by modeling as a max flow problem. To continue getting more practice with modeling real world applications with network flows, you have the opportunity to investigate the fair rideshare/carpool problem. The problem is written as follows:

You and three of your friends decided to live together in San Francisco after graduation. There is only one car available to share among all of you for commute . Each week, you and your friends work out a schedule like the following to announce which days each of you will be using the carpool.

	M	T	W	R	F
Person 1	✓	✓	✓		
Person 2	✓		✓		
Person 3	✓	✓	✓	✓	✓
Person 4		✓	✓	✓	✓

Besides coming up with the carpool schedule, you would also like to come up with a way to allocate the driving responsibilities fairly. Here is one way to define fairness. Let the people be labled P = {p1, p2, p3, p4}. We say that the total driving obligation of a person over the 5 days is the **expected number** of times that this person would have driven, had a driver been chosen **uniformly at random** from among the people going to work each day.

Question:

How would you come up with a fair allocation of driving responsibility using network flows for any arbitrary carpool schedule of 4 people over 5 days?

List of resources

- Network Flows Algorithms Textbook - Chapter 2.2 walks through the explanation and proof of modeling the fair carpool sharing problem as the max flow problem. If you choose to use it as hints, use a different driving schedule in your example.
- Another proof of using max flow to model fair carpool sharing - The first problem

Things we would expect you to accomplish to have a successful project

- What would the diagram of the network flow look like for an arbitrary driving schedule? Explain what the vertices and edges represent and explain what the capacity and flow on each edge mean.
- Explain/Prove how you would allocate the driving responsibility using finding max flow of the corresponding network.
- Implement the Edmond Karp Algorithm to determine the driving responsibility of a given driving schedule with test cases.

Idea 6: Timetabling with Genetic Algorithms

In the last lecture, we introduced genetic algorithms as another approach to estimate solutions to NP-hard optimization problems. One important application of genetic algorithms is the task scheduling or timetabling problem. For example, if you are the registrar at Olin trying to come up with the course schedule for Fall 2021, you need to consider various constraints including limited time slots during a day, professor availability, classes that are commonly taken together during the same semester and classroom location. The goal of the timetabling problem is to minimize conflict in all these aspects. For the final project, you are welcome to explore and develop a course/task

scheduling program using genetic algorithms. How many constraints you consider for each task or course is up to you. You can be creative with the econding, crossover and mutation steps of your program.

List of resources

- Simple example of using genetic algorithms to schedule timetables
- Paper on Solving the Modular Example Scheduling Problem with Genetic Algorithms
- Paper on Genetic Algorithm for University Course Timetabling Problem
- Code and explanation on Modular Exam Scheduling Problem with Genetic Algorithms
- Code and Documentation of a task scheduling on computer processors with genetic algorithm

Things we would expect you to accomplish to have a successful project

- Explain how you encode each individual task or class with its properties such as location, professor, time slot, day of the week etc.
- Explain what your fitness function is
- Explain the selection procedure for deciding which individual goes into the mating pool
- Explain your crossover method
- Explain your mutation method
- Implementation of your task scheduling/timetabling program

Idea 7: Max Cut Algorithm

“A Max Cut algorithm in between random and the SDP that uses eigenvectors.” - Alice

List of resources

- Trevisan paper: https://arxiv.org/abs/0806.1978
- DeSoto improvement of the analysis: https://arxiv.org/abs/0910.0504

If you want to pursue this idea, reach out to Alice (alicejpaul@gmail.com) with questions when writing your project proposal. In your proposal, please also include:

- Short project description
- List of at least 3 resources that you will investigate
- A breakdown of what you will accomplish in order for us to evaluate your project as successful

Idea 8: Online Algorithm for Submodular Maximization

“A neat online algorithm for submodular maximization. It is a very intuitive and simple algorithm so would be fairly approachable I think.” - Alice

List of resources

- https://theory.epfl.ch/moranfe/Publications/FOCS2012.pdf

If you want to pursue this idea, reach out to Alice (alicejpaul@gmail.com) with questions when writing your project proposal. In your proposal, please also include:

- Short project description
- List of at least 3 resources that you will investigate
- A breakdown of what you will accomplish in order for us to evaluate your project as successful