CIS 1890 Final Project Description

Project Overview

The goal of the final project is to give you experience with applying the tools and concepts we've learned to a problem of your choosing. **Projects must be completed in pairs of two students,** although I am willing to make exceptions for extenuating circumstances and odd numbers. The requirements are open-ended and you are free to choose a project that is more theoretical, experimental, applied, etc. based on personal preference. Ultimately, all groups should produce a project of roughly equivalent scope.

Please see the **Project Ideas** section at the bottom for some inspiration.

Project Timeline and Grading Breakdown

You cannot use late submissions for any of the final project deadlines.

- **Mar 21** — Project partners due
  - 3% of grade
- **Mar 23** — Project proposal due
  - 10% of grade
- **Apr 18** — Project check-in due
  - 10% of grade
- **Apr 25** — Project presentations (in class)
  - 25% of grade
- A**pr 25** — Final project submission due
  - 52% of grade

Choosing a Project

While you and your partner have near-total freedom in selecting a topic for your project, it must be clearly related to the overarching theme and material of the course. Although I encourage you to use Python and PicoSAT/OR-Tools, you are welcome to use other tools if there is a good reason.

See the Project Ideas section at the end of this document if you need some inspiration. Most of your projects will likely fall into one or more of the following rough categories:

- **Applications.** A more applied project seeks to utilize the tools we have learned in order to effectively model and solve a challenging problem. For an applied project, you should imagine that you are building a full product, rather than just a basic script to solve one instance of the problem (which is too small of a scope). You should implement more user-friendly features, such as: a simple command line interface, GUI, or web app; parsing problems from input files; writing solutions to output files; solving multiple varieties of the problem or allowing customization of constraints; etc. Often, an applied project will involve OR-Tools rather than a SAT solver.

- **Theory.** A theoretical project seeks to expand our knowledge about some of the high-level techniques we've learned, or even develop new techniques. Such a project might be more implementation-based or have a larger written paper component. You also might choose to read relevant published papers and compare them, expand upon them, apply them to new domains, try to reproduce their results, etc. A theoretical project may be more likely to involve a SAT solver or code written from scratch rather than OR-Tools.

- **Experimentation.** An experimental project seeks to compare or highlight a technique or tool by rigorous testing and the scientific method. Rather than being a stand-alone category, most experimental projects will likely lean towards being more applied or theoretical. An applied experimental project might, for example, involve testing and comparing multiple tools and attempting to empirically classify which inputs are solved easily or with difficulty. A theoretical experimental project might, for example, involve modifying or improving a technique and measuring the effect in a variety of situations.

Important: Even if your project is not primarily experimental, you should still be doing a good amount of experimentation and testing in your project. I will want to hear about your experimentation in your final submission. For example, if you are modeling a problem in OR-Tools, test multiple different ways to write constraints and evaluate which performs best. If you are implementing a technique we learned, test to see how much improvement it offers over some baseline.

Remember, it's OK if your project does not end up giving an improvement over the baseline, as long as it was a reasonable idea. I would much rather see a project that discredits an interesting proposal, ideally with some analysis of why it may not have worked, rather than a project whose results cannot be reproduced or which is successful only on a couple cherry-picked examples.

Partner Selection (due Mar 21 by 4pm)

You should find a partner for your project, using the *People* page on Canvas and the *Search for teammates* page on Piazza. You are also welcome to shoot me an email and I can try to match you to someone else.

Once you have found a partner, simply submit a blank page (or your favorite meme) on Gradescope as a group submission with your partner and yourself.

If you would prefer to work alone due to extenuating circumstances, please reach out to me **as soon as possible** to discuss. Also reach out to me very early if you would like to have a group of 3 and I will consider it.

Project Proposal (due Mar 23 by 4pm)

You and your partner must jointly submit one project proposal document containing the following information:

- The names of both group members
- A title for your project
- A 1 to 3 paragraph description of the project — what is it? What are the goals of the project? How does it relate to the course material?
- A 1 to 3 paragraph description of your tentative plan for implementing the project — what tools or techniques will you apply? What learning resources will you use to guide you?
- A 1 to 3 paragraph description of how your project should be evaluated — how will you test it? What will be your metrics for success?

The project proposal must be uploaded to Gradescope as a group submission with both partners.

Project Check-in (due Apr 18 by 4pm)

At this point, you should have implemented a significant amount of the functionality of your project. As an estimate, you should feel that your project is at least roughly 75% complete based on your own understanding of the project's goals and evaluation metrics (from experience, you are usually never as close to “done” as you think you are).

You and your partner must jointly submit one project check-in document containing the following information, which can be any reasonable length as long as it answers all the criteria:

- The names of both group members
- A title for your project
- An overview of what you have accomplished so far — if possible, include data, results, screenshots, etc. No code files please (short snippets are OK).
- An explanation of how the scope and goals of your project may have been revised, and why
- An overview of what you still have left to accomplish, and your path to doing so

The project check-in must be uploaded to Gradescope as a group submission with both partners.

Project Presentation (due Apr 25 in class)

You and your partner will jointly give a 5-minute presentation to the class demonstrating the functionality or end-result of your project. In addition to the 5 minutes, you will have 2 minutes afterwards to take questions from the class about your project.

How you give your presentation is up to you. Given the limited time allotted to you, I recommend that you focus on overview, features and functionality, and/or results, rather than diving very deep into implementation details. You may want to consider having a live demo and/or making a few slides explaining your project.

Final Project Submission (due Apr 25, by midnight)

You and your partner must jointly submit the following **in a zipped folder** named `189Project_Firstname1_Firstname2.zip`, uploaded to Gradescope as a group submission:

- All code and notebooks related to your final project
    - I should be able to run your testing/experiments!
    - Make sure to include all data files relevant to your project, if applicable
- README document explaining:

    - An overview of what you have completed in the final project
    - An overview of the file structure / explanation of what each file contains
    - Any instructions necessary for running your final project, including running any tests, installing any necessary dependencies, etc.
- RESULTS document, if applicable to your project, containing a summary and explanation of results from testing or experimentation
    - All results should be reproducible in the code/notebooks you have provided, but you should still compile them in this document.
- Any other files that you feel add necessary context for your project

There are no explicit requirements for the length of any of your documents; they should be long enough to include all the important information, but not unnecessarily long.

Project Ideas

This is a non-exhaustive list of ideas that you are welcome to take for your project or simply use for inspiration. I have additional details, resources, and suggestions for many of them, so please reach out if you are interested!

- Pick one or more of the problems on CSPLib, a public database of constraint programming problems, and attempt to model and solve it effectively in an applied or experimental project.
- Pick a different puzzle game than Sudoku and build an application to play it using SAT, MIP, or CP in an applied project.
- Formulate a real-world scenario as packing, planning, or scheduling problem and build an application to solve it using SAT, MIP, or CP in an applied project.
- Pose your own interesting CP or MIP problem, attempt to model and solve it effectively, and investigate experimentally what makes an instance of the problem hard or easy.
- Build a basic package manager using SAT or CP for dependency resolution (you may also want to take a look at the CUDF format for package management).
- Use constraint programming for procedural content generation of 2D maps by solving constraints about which tiles can appear next to each other (here's a cool blog post).
- Build a visualization tool that integrates with PennSAT to allow the user to run the solver step-by-step and view the intermediate search tree, solver state, etc.

- Invent or modify, implement, play around with, and rigorously test a variety of decision heuristics to see how they affect the performance of PennSAT in an experimental project.
- Expand PennSAT to use some of the modern SAT solving techniques we learned, in particular 2-Watched Literals and Conflict Driven Clause Learning (CDCL). Just 2WL + CDCL would definitely be large enough in scope — this might be a tricky project, so please reach out to me for advice and resources if you're considering it.
- Build your own CP solver inspired with lazy clause generation using PicoSAT as a backend.
- Use the CryptoMiniSAT solver to implement an attack on a cryptographic cypher (this may be difficult if you have no background in cryptography).

Disclaimer: It's possible that some of these projects are more difficult than I have estimated; reach out to me or do a bit of preliminary research to make sure that you have some idea of where to begin before committing to a project.