CPSC 433: Artificial Intelligence - Project Description

(Version 1)

There will be one big team assignment, namely

1. developing (writing an abstract formal proposal) and
2. implementing (creating code for)

a search system for a instructor provided application area.

The application area will involve optimization of a constraint satisfaction problem (CSP).

In the chosen CSP problem, different events/objects must be fit to available openings.

- Valid solutions require the event/objects to fulfill hard constraints that limit where they can be placed.
- Optimal solutions fulfill the hard constraints (true/false requirements), while further placing events/objects in the best location to maximize/minimize soft constraints (countable values) in the most preferred manner.

Team Size

We prefer teams of 5, but 6 can be approved.

The project is generally able to be completed by as we as 3 students but 4 is a good minimal size for workload in a semester. 5/6 allows for a surprise attrition through-out semester without project becoming infeasible to complete. Groups that are too small after midterm/proposal points of semester will be merged to complete the coding implantation as a combined group.

Teams can be cross-tutorial and formed by students themselves.

If you don't have a team when I request groups to inform me of their groups name, email addresses, and UCID, then I will assign a team to you. Generally this team will have a focus on group members within tutorials. Tutorials represent one opportunity to communicate with you team members on planning during the semester.

Search Paradigms

As will become obvious during the course, for each application area there are different search paradigms that can be used to develop a search system solving instances of the problem. (Set-based, or-tree-based, and-tree-based are high level examples of these with a number of sub-variants within them.) Each paradigm leaves a lot of room for instantiations, based on the knowledge you have about the particular application area.

Assessments

The assignment is intended to make you aware of different paradigms and the possibilities regarding their instantiations. Therefore,

1. You will write a paper to propose two abstract solutions to the application area problem using two different search paradigms
2. Then I will choose one of these two for your group to implement in code. (I generally select the paradigm that is best communicated and gives me the most confidence your group can complete the code for a well functioning solution.)
3. You will demonstrate the running of this system at the end of the semester to me.

More precisely, in the proposal paper I expect each team to:

1. **Present two different search models** (using different data structures as states and providing the necessary information how to instantiate these data structures)
2. **Describe them precisely** (i.e. in a mathematical manner similar to what I use in the lectures, concentrating on the necessary information for the instantiation), and present at least one sensible search control for each model to complete the definition of the search processes your search system might employ.
3. **Demonstrate what your search processes will do by applying them to a small search instance chosen by you.**
4. In addition to formal definitions, I also expect a **natural language explanation of the models, controls and processes and the reasons** why you have chosen the particular models.

Both paper and the final system will be assigned a letter grade based on its quality.

Optional Proposal Instructor Draft Meetings

Regarding the paper, it is a good idea to crosscheck with me your models, controls and so on, before you submit the paper. In fact, I will be making myself available to have a meeting with each team before the team submits the paper!

Non-functional Requirements

Your paper, again, can be produced by any text processor you want to use, but I expect that you send me pdf-files that are readable everywhere. Note that using Word can result in difficulties around formulas and sometimes fonts! LaTeX can generally be used to better success with formulas if you are familiar with it.

You can use whatever additional tools you want, and you can develop on whatever machine you like**, as long as the final system is written in language that can be easily deployed to run on one of our Linux machines in the undergrad lab** (resp. one of our Linux compute servers), as a standalone application. Your system can not use internet based resources or AI systems implemented either locally or remotely (existing AI APIs or things like LLMs).

The most common language choices are Python, and Java as all students will have taken a course in these two. Naturally, both of these languages have some performance challenges with being compiled interpreted languages. However, both are more than capable of the scale needed for the problem given in this course if the search paradigm is designed correctly. C++ is the most common third choice and does come with performance benefits. Depending on your groups experience you may find debugging C++ when they go wrong in your complex code more challenging to do than the prior languages.

Each team will have to give a presentation of their system in the last week of classes and the grade of a team for the system will be mainly based on how the system fares during verification tests I bring with me to the lab during this demonstration/presentation.

After the initial validation tests have been run once and pass/fail noted teams are given the remainder of the demonstration time (one hour) to fix any bugs they think they may have. If these bugs are fixed and the systems pass the verification tests after the fixes, then full or partial credit for the test may be given to the group. Note, these fixes must be general code chances and not just attempts to hard code the desired test output.

I expect each team to fix all problems that the system shows in the demonstration before the final submission of their code later that is used on the LARGE INPUT examples for final testing. Not completing these fixes and having faults in the LARGE INPUT solutions result in penalty adjustments to the grade given during the demo verification tests of the system done prior.