Programming Project, Part 2
CS 5260, Spring 2023

In Part 1, you were required to adhere to a somewhat specific specification, albeit with lots of room for research and creativity in the system that you designed, implemented, and evaluated. In Part 2, you are given considerably more leeway in how you expand, improve, and otherwise alter the scheduler/simulator of Part 1. Nonetheless, you might still be be trying to successfully complete Part 1 specifications.

Your focus in Part 2 can be on ***one or more*** of the following:

1. In the case where you did not get your Part 1 agent working or evaluated, you can keep working on the code from Part 1. But you should attempt to do more, because a lack of progress using this option alone will not reflect well in your Part 2 grade!
2. Assuming you were able to satisfy the Part 1 evaluation criteria, you can improve on your own project code, perhaps replacing a computationally expensive search that really limits the number and quality of schedules derived (e.g., a beam search) with a less expensive search that 1) still yields schedules of acceptable quality and deeper depths, or 2) executes its search much more quickly. If you use this option, repeat your experimental evaluations from Part 1 to show all differences and improvements.
3. Ideally, you will be at a point where you can start expanding the Part 1 specification in at least one of several ways. For instance, gamifying the simulation or learning macros are two potential options. It is likely and intended that this option (expanding on Part 1 to include other functionality) will be the focus of most of Part 2.

Knowledge sharing guidelines will also change in Part 2. You can release your Part 1 code in totality (free revealing), and others can use all or part of it in their Part 2 designs and implementations, even to the exclusion of their own Part 1 code. Thus, there is the potential for a 'soft reset' for someone who uses another person's code. If you use another's *released* code, then you must cite that reuse when referencing it, and you must declare the reuse in your final deliverable (an honor code requirement). The person's whose code is reused will get significant points for the reuse by others of their code. *The better documented your code and the better it performs, the better chance that your code will be reused by others.*

It is possible that you will use another student's released code if you did not do well in Part 1, or you think another student's code would be a better jumping-off point for Part 2, or if you see advantages to one code base when jointly developing Part 2 enhancements with one or more other students. You might also ask whether it would be advantageous to "start over" on Part 2, rather than further altering your Part 1 code. Whether this is a good option or not depends on how far away you were from creating a performant agent in Part 1, and how far you will get with Part 2 enhancements when building on another project. Please ask if you want feedback in deciding.

In addition to code sharing through free revealing, knowledge brokering possibilities abound. If, for example, there are several students interested in implementing a game and an associated game manager, they can cooperate on the creation of the game manager, a communication protocol between players and the manager, round robin infrastructure, etc, splitting the development responsibilities as they see fit. Whether through knowledge brokering or not, the development efforts in Part 2 can again be free revealed, so that other students can take part in any general enhancements created by another student.

Project Part 2 Ideas

1. Gamification

The AI you develop in Part 1 enables a virtual country to plan for possible futures, and to quantify the utilities of each of those possible futures and the plans/schedules that lead to them. Each of these futures undoubtedly involves other virtual countries, and that future could only be realized if the other countries did what the self-country's plan called those other countries to do. But in Part 1, there is no acting on the plans that a country develops. Part 1 is about imagining a future rather than realizing a future, akin to the way that minimax search in chess explores the implications of moves without actually making a move.

In Part 2, one direction you can go is to embed your simulations into game players, perhaps coordinating with other students, to build a game that is played by AIs rather than being played by humans. In other words, the schedules developed in Part 1 would be executed in Part 2 and actually change the common, virtual world state.

To create a game of AI players in a virtual world, the AI players must each see the same world state, and when a transform or transfer operator is actually executed by any player's plan, the single global world state is changed, presumably by an omnipotent game manager.

The players must be able to communicate through a shared protocol. Consider the following as a potential start:

(PROPOSE c_i S S_{id} c_k — c_k in S) where S is a schedule, S_{id} is a unique id, c_i and the c_ks are countries, and the c_ks are referenced in one or more operations of the schedule S.

The country c_i, which developed S, submits S to a central "blackboard" that all countries in the world can see, or perhaps only the c_ks can see. If a c_k country agrees to the schedule, it communicates this to c_i, again perhaps through a central blackboard.

(AGREE c_k S_{id}), indicating that c_k agrees to S_{id} (from which c_i can be determined).

If all c_k countries AGREE, the game manager executes the operators in S, presumably in sequence, updating the world state after each operation.

This is just the start of a possible protocol, and if this direction is selected, then the major work would be in defining a game manager that (1) moderates communications between countries through a shared "blackboard" that can be accessed by each country (asynchronously?), and (2) executes agreed-upon plans and changes the world state appropriately.

2. Macro Learning

A macro is a sequence of two or more operators that are conjoined for execution as a package. In principle, a complete grounded schedule can be a macro, but the most useful macros are broadly applicable and often only a few operators long. Typically in a macro, there are variables so that the macro can be used for different situations; in our case, different countries, resources, or quantities of resources. For example, consider a "free-trade" macro which might be only two operators (both Transfers) long. One country c_i transfers a resource to another country, c_k, and then c_k transfers another resource to c_i in exchange. "Trades" of resources are mutually beneficial and thus would be frequently included in schedules. The goal would be to learn this and other macros that proved to be generally helpful in reducing search costs.

3. World Modeling

You may significantly expand modeling of the virtual world, perhaps remaining in the confines of the Transfer and Transform operators, or perhaps defining additional operators. Expansion can include modeling recycling, reuse, decay, and/or regeneration of resources over time, limited or all-out war (which need not be part of a gamification involving a game manager), natural catastrophes, or non-uniform resource weighting among countries. Additionally, indirect processes can also be modeled, such as the inclusion of "money" as a resource to mimic financial and economic processes.

4. Adversarial Search

In Part 1, all adversarial aspects of trading were handled by bundling the probabilities of the success of any given schedule into the Expected Utility value. While this is a very useful tool for planning-based agents, it somewhat abstracts away the reality that, in the real world, the planned actions of our agents will rarely go exactly according to schedule. As such, it may be useful to carry out our search either using an exhaustive adversarial strategy such as Minimax with Alpha-Beta Pruning or using a probabilistic strategy such as Expectiminimax. In both cases, search would not relentlessly march to a solution that is simply the best for us, it would also search over what is best for other countries (perhaps limiting ourselves to 2 countries in this case). This would grant our agent a higher degree of flexibility in choosing successive actions when its schedule does not proceed exactly as planned.

5. Student Suggestions from Piazza

There were a number of fantastic ideas put forth on Piazza that would make for good Part 2 projects. Some of these ideas include: Decaying Resources, adding the concept of "Maintenance" or "Upgrade" actions, Research Trees, Resource Tiering, Country Reputations with causal effects on the probability of other countries accepting deals, etc. I highly suggest that everyone look over the relevant Piazza thread on these topics.

Project Part 2 Deliverables

The primary narrative of project Part 2 will be in the form of PowerPoint slides and a video presentation using those slides. You will also submit well-documented source code. Your deliverables should include:

- **TalkSlides.pptx:** PowerPoint slides to accompany your video
- **VideoLink.txt:** Text file containing a link to your narrative video (e.g., a private link on YouTube or a link to a Zoom recording of no more than 15 minutes). The bulk of your Part 2 grade will be based on the video and slides, with expected content to include:
  - A figure and explanation of the comprehensive architectural design of your system,
  - A focus on any improvements, additions, changes, or expansions that you made since Part 1,
  - Summaries of experimental studies, including explanations of selected test cases, graphs, and tables,
  - Comparisons between the results of your Part 1 and Part 2 outputs, including the quality of the resulting schedules, the speed or efficiency of your algorithms, and the complexity of the resulting outputs,
  - Citations to outside sources that you used, including references to other students.
- **SourceCode.zip:** well-documented and well-formatted code, including:
  - **README:** Containing clear and detailed directions for running your code and identifying the file containing the `country_scheduler` function, and any other important top-level functions.
  - ***You will also be graded on the documentation and format of the entirety of Part 1 and Part 2 code.***