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.

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.***

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.

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.

CS 5260: Intro to Artificial Intelligence

Week 10

Will Hedgecock, Ph.D.

Programming Project, Part 1 Wrap-Up

- Reiteration of purpose:

To help you gain practical experience thinking about how real-world problems might be specified, quantified, and represented for reasoning in an AI-based agent

- Concepts introduced:
  - Heuristics
  - State Utilities
  - State Representations
  - Action Representations
  - Basic Search Strategies
  - Probabilistic Encoding

- Questions/concerns?
- Video creation/submission issues?

Programming Project, Part 2 Kick-Off

- Purpose:
  - Explore *at least one* additional AI concept using your Part 1 results as a basis
  - Compare how different strategies, methods, and assumptions affect the apparent intelligence of your agent
  - Investigate how agents can be expanded and iteratively improved to increase complexity and intelligence over time

- Non-Purpose:
  - Reinvent or reimplement your Part 1 code
  - Perfect or improve your existing agent *from a programmatic viewpoint*
  - Compete with your classmates on creating the most advanced agent
  - Accumulate and hoard knowledge

Programming Project, Part 2 Kick-Off

- Three Options:

  1. If your Part 1 code was not working satisfactorily, continue working on it **but increase the difficulty and explore new concepts**

  2. Improve your Part 1 agent, replacing computationally expensive search strategies with less expensive strategies that yield better schedules, deeper searches, or faster execution times. Your improvements must be somewhat rigorous and more than surface-level deep (i.e., exploring breadth-first search instead of depth-first search won’t cut it)

  3. Expand the Part 1 specification itself to investigate more advanced concepts such as macro usage, gamification, resource depreciation, recycling, etc.

Programming Project, Part 2 Kick-Off

- Guidelines:
  - Knowledge sharing is ***expressly allowed***
    - You can release your Part 1 code in its entirety
    - You can use someone else's Part 1 code in its entirety
    - Extra credit if others re-use parts of your released code
  - Collaboration between students on a unified product item ***is allowed***
    - Creation of a "game manager"
    - Creation of communications protocols
  - Format and goals are much more under-specified
    - Pick a topic that is interesting *to you*

Programming Project, Part 2 Ideas

Gamification

The AI you developed 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 a common, virtual world state.

Programming Project, Part 2 Ideas

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 a resource back 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.

Programming Project, Part 2 Ideas

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

Programming Project, Part 2 Ideas

Monetization

A glaring omission in our Part 1 trading algorithms is that there is no concept of an *indirect resource* such as money. The inclusion of such a resource would open the door for much more complex modeling of how trade actually works in the real world, providing for clever ways of offloading manufactured waste, as well as allowing for more “realistic” state quality functions, such as GDP and debt.

In order to model money, it may be possible to add a new Purchase operator which specifically operates on the exchange of resources for money, or we could use existing Transfer operators. Additionally, money could have a devaluation effect over time (perhaps using the schedule depth as a proxy for time) to mimic inflation, so that you need more of it as you progress in order for longer schedules to maintain a desirable state quality value.

Programming Project, Part 2 Ideas

Resource Decay, Maintenance, and Recycling

Credit: John Ford

What would it look like to introduce a sense of entropy into the game? Manufactured goods don’t typically last forever, and there is often a balance of maintaining existing vs. creating new goods.

For a concrete example, consider the quality of a house just built vs. one built 50 years ago. If no resources have been put into maintaining the older house, it may be so far gone as to be uninhabitable.

The simplest approach here might be to do the opposite of renewables and have a decay rate applied to certain resource stacks every game tick or “turn.” However, it would be nice to find an elegant way of allowing for some sort of *Maintain* or even *Upgrade* transformation that would have a cost, but likely a lower cost than the cost for building *new*.

Programming Project, Part 2 Ideas

Research Trees

Credit: John Ford

Several popular strategy games introduce the concept of a *Research Tree*, in which you spend some resource(s) and/or currency now to unlock the ability to build something better in the future. The overall concept is fun to consider from the perspective of an AI, because just like humans, depending on the search strategy (perspective), the best long-term utility of such a choice might not be considered.

There are a few ways to model this just using the resource and transformation templates and no other mechanic changes. A new resource of “research credits” is created. Transforming combinations of raw and even other manufactured resources could produce research credits. At certain credit thresholds, new transformation templates become available that require the credits as input and also return the credits as output, thus not changing the current research “level.”

For more complex research trees, intermediary “blueprint” resources could be introduced that require research credits as inputs to generate. They would then be inputs into subsequent specialized transformations that would similarly return the blueprint as an output so as not to take away “knowledge.”

Programming Project, Part 2 Ideas

Research Tiering

Credit: John Ford

While often paired with research, resource tiering can be done with or without that concept. In the real world, we don’t typically produce one type of something. Manufactured goods have varying quality levels, or tiers.

Housing is one area where this could be complex and interesting. While the game of Monopoly provides a simple example of transitioning, houses to hotels, what could be represented in our country models? Could a House I, II, and III represent increased population satisfaction at a cost of increased square footage (available land)? Could the introduction of High-Density Residentials be used to offset this?

Tiering could be done by simply having a tiered transformation require the preceding tier as an input, e.g. House II transformation requires House I. For more complex modeling, it could be tied to the Research Tree concepts, requiring research as an input instead of a prior tier. It could also be associated with the resource decay, where higher tiers decay at a slower rate than lower tiers.

Programming Project, Part 2 Ideas

Reputation as a Resource

Credit: John Ford

This particular idea feels like a natural extension of the game's premise centering around countries trading amongst each other. Soft power exists on the world stage. It's less concrete than hard power (resources), but it has subtle implications for how countries interact.

A reputation resource would probably need to be done as a signed number, representing both negative and positive influence. It could be tracked as a single number per country, representing more of a binary outlook on the countries of the world as either good or evil, e.g., UKRAINE has REP 5, RUSSIA has rep -1. It would probably make more sense to track it as a separate resource for each country, e.g., USA has REP_UKR of 2, while RUSSIA has REP_UKR of -2.

The reputation resource would then have an impact on probabilities. A country might be slightly more inclined to accept a sub-par (utility) deal. A country might be more likely to declare war. There could also be new transfers or transformations to represent various aid (financial, military, medical, etc), and these would only provide the initiating country with reputation in return.

Programming Project, Part 2 Ideas

Relative Advantages

Credit: Peter Callahan

Since countries do not have similar capabilities in the real world, being able to measure the impact of an additional capability would be a realistic application of this kind of model.

Consider this example: Run the trade simulation from the perspective of a country whose capabilities completely match those of its neighbors. In a subsequent run, add an additional transformation to the source country (or one of the non-self countries) and see how this asymmetry changes the model output. Are countries better off with high-tech neighbors or only if they possess the technology themselves?

Alternatively, two countries could have similar transformations with one country having a slight efficiency advantage in terms of resource consumption. How would this comparative advantage change the outcome of the simulation?

CS 5260: Intro to Artificial Intelligence

Week 12

Will Hedgecock, Ph.D.

Programming Project, Part I Issues

- Bravo to EVERYONE on their submissions and presentations

- **Common Issues**:
  - Using resource weights to nudge the AI in a specific direction
    - Weighting should be used primarily to indicate “intrinsic value” of resources
    - E.g., Does having metallic alloys actually make the people in your country happy, or is having them simply a stepping stone to creating Electronics?
  - Using the wrong frontier container for your specified search strategy
    - E.g., You **cannot** use a Priority Queue with Breadth-First Search
  - Including non-participatory countries in the computation of the probability of success of a schedule
  - Using dynamic weights with discontinuities
    - Not only does this introduce human intuition, it also makes your EU values “jump” in unexpected ways

Programming Project, Part I Issues

- **Common Issues**:
  - Calculating rewards based on successive states instead of based on the current state relative to the very first initial state
    - Taking actions results in “partial schedules”, “full schedule” is only found at leaf nodes which will always have a number of actions equal to the max_depth
  - Artificially creating an “explicit graph” before examining any of the nodes
    - Especially problematic in depth-first, best-first, or heuristic-based searches
    - Cuts away HUGE portions of the search space
  - Simply returning schedules found at max_depth instead of storing them somewhere else so that you can return the “best” schedule found
  - Ordering actions and operations such that the search space is always explored in exactly the same manner, especially problematic with naïve search strategies
  - Decreasing EUs
    - More commonplace than expected with no common or obvious reason

Programming Project, Part I Comments

- Looks like most people used a high Discount Reward constant gamma
  - Resulted in more aggressive/risky transfers that almost always paid off
- Benefit of “anytime” search algorithms is that search continues even after a solution is found
  - In our case with a max_depth, agent would restart with a depth of 0 after finishing its search and iterate over your algorithm forever, potentially finding better solutions in subsequent iterations
- Don’t forget “reached” lists
  - May dramatically speed up your search, improve results, and keep you from getting into Transfer loops
- Storage of num_output_schedules best schedules in a Priority Queue allows you to keep track of the best solutions without you having to do anything complicated
- Another way to prune the state space is to ignore states (not add to the frontier) that have EUs so low as to be unlikely to be useful going forward
  - Perhaps as a multi-order-of-magnitude decrease from other schedules at same depth

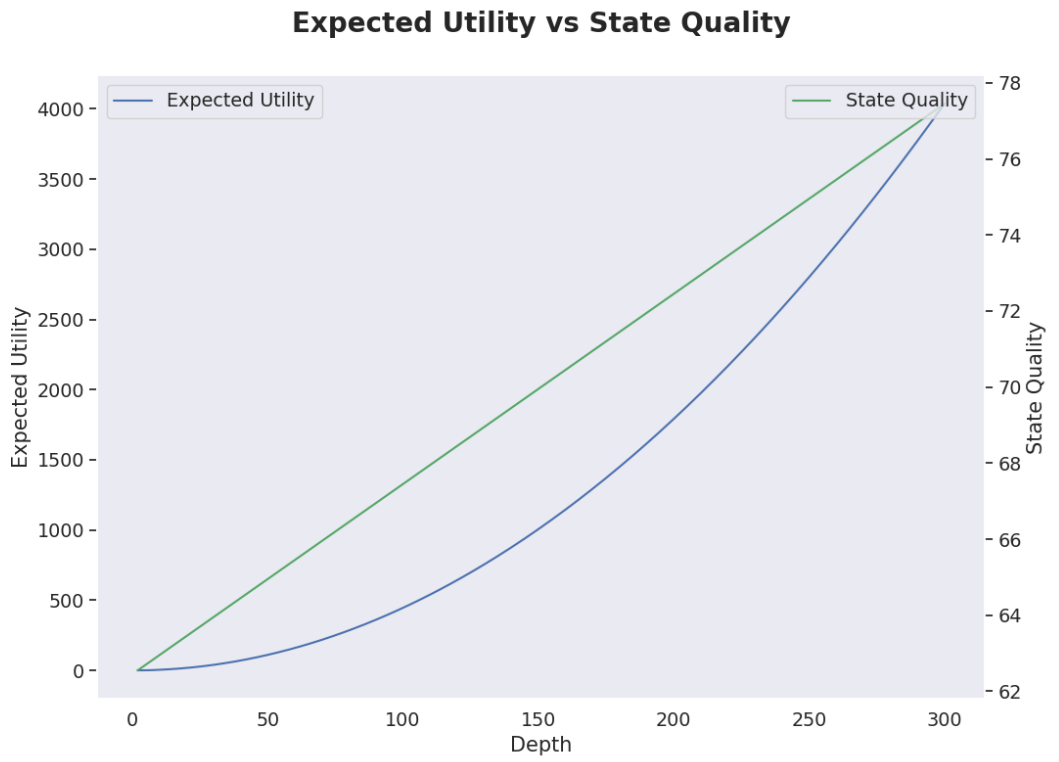
Programming Project, Part I Comments

- **Interesting Results**:
  - Everyone’s individual State Quality functions were extremely diverse and very interesting
    - Average US household size = 2.5, and average number of electronics = 13.4
    - Weighted resources based on total world amount (scarcity) instead of population
  - People’s choices of starting states for the world were equally creative
    - Poor vs rich, balanced vs imbalanced
    - Using Fallout 2 (postapocalyptic world with no currency and extensive bartering)
  - Several students computed actual state space size and showed how intractable some generic strategies can be
  - Interesting ideas for Part 2:
    - Creation of a “well” resource to allow for renewable water
    - Devaluation of resource weights over time (fiat money)
    - Implement A* search where “g” is the Undiscounted Reward

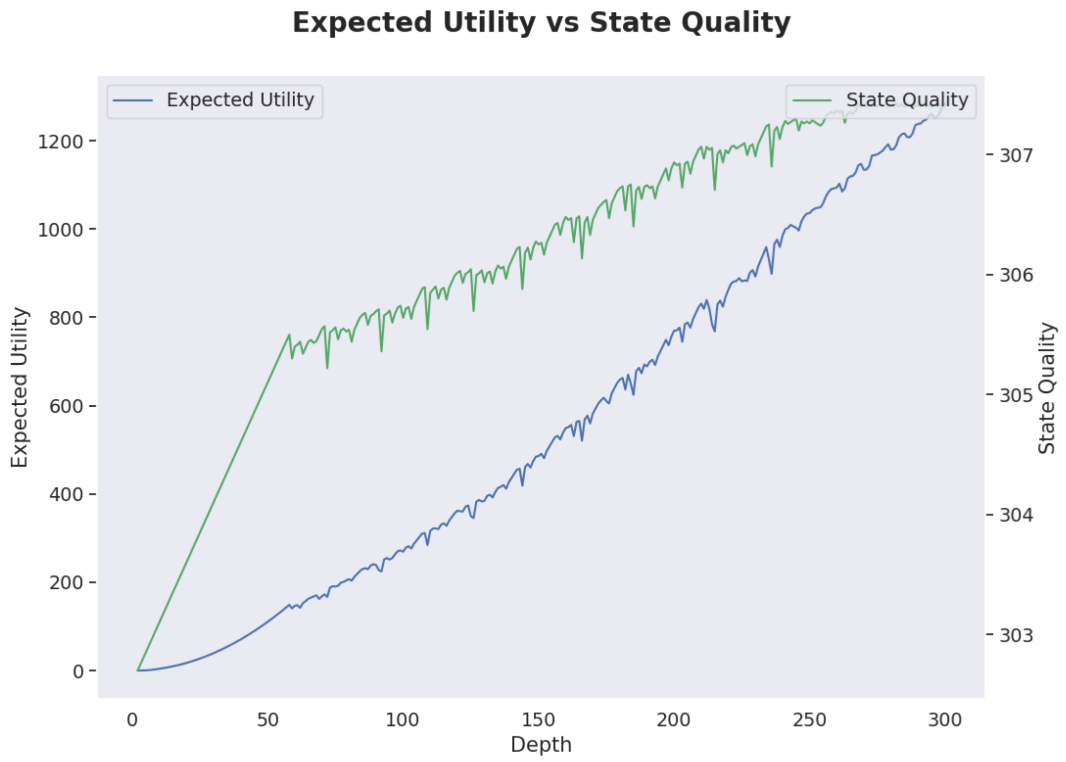
Programming Project, Part I Comments

- **Interesting Results**:
  - Peter Callahan experimented with varying the number of “resource multipliers” available for an action
    - Found that increasing the multipliers increased the branching factor but also significantly improved the EUs that the AI was able to find
  - When comparing the results of a poor vs. a rich country, John Ford found that the poor country was able to improve its status at a rate of only ~20% that of a rich country
    - In almost all cases, a country was most likely to first transform its existing resources into higher-valued resources before turning to trade to increase its EU
    - It was necessary to take multiple steps backward in order to make progress to better and better states
  - Result graphs for 3 varying starting states:

Programming Project, Part I Comments

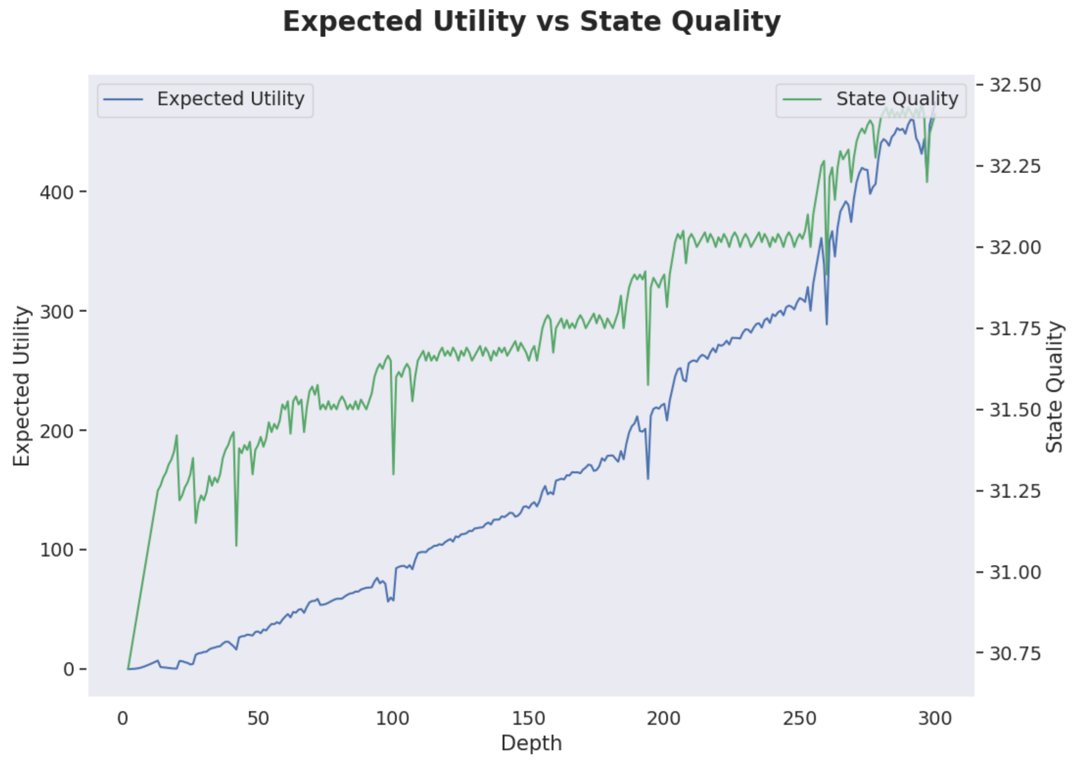


Initial State: All countries rich, able to transform lots of natural resources to improve quality



Initial State: Very imbalanced set of natural resources for each country, can't transform forever

Programming Project, Part I Comments



Initial State: All countries poor, very few transformations able to be carried out initially