Programming Project

CS 5260, Spring 2023

Overview

The CS 5260 programming project is somewhat open-ended, allowing you to participate in the specification process, including design, implementation, and evaluation. The project requires integration of knowledge and skills learned in earlier CS courses and professional experiences, and you should ideally incorporate knowledge from other disciplines and areas of interest to you.

The goal of the project is to build a multi-method AI agent that plans and models trading and development (e.g., farming, manufacturing) decisions for a set of virtual countries, with each country possessing virtual resources and incentives for cooperating and competing with other countries. The project is split into two parts. In Part 1, you will create a well-defined backbone for an AI agent (a.k.a. virtual country) that simulates potential actions in a simulated world; however, the design specification for Part 1 still has gaps that you must define. In Part 2, you will build additional functionality into your AI agent from Part 1 as you see fit (e.g., for learning, game play, game definition, modeling war, reasoning under uncertainty, etc.).

Project Background

A world containing M virtual resources is divided among N virtual countries, each with some amount (or none) of each resource.

Basic Operations

The are two families of operations: TRANSFORM operations that manipulate resources within a single country and TRANSFER operations that manipulate resources between countries.

Transformations

TRANSFORM operations (a.k.a. Transformations) represent how a country (C) transforms one set of resources into another set of resources (e.g., by manufacturing, agriculture, internal strife, etc.). The resources that comprise the INPUTS and the OUTPUTS are allowed to overlap, indicating that a given input is not completely consumed, or is even increased by the TRANSFORM. All transformations must abide by pre-established templates that outline the unit number of inputs needed to create a unit number of outputs:

Housing Template

```
(TRANSFORM C
           (INPUTS (AvailableLand 1)
                   (Population 5)
                   (Water 5)
                   (MetallicElements 1)
                   (Timber 5)
                   (MetallicAlloys 3)
                   (PotentialEnergyUsable 5))
           (OUTPUTS (Housing 1)
                    (HousingWaste 1)
                    (Population 5)
                    (Water 4)))
```

Alloys Template

```
(TRANSFORM C
           (INPUTS (Population 1)
                   (MetallicElements 2)
                   (PotentialEnergyUsable 3)
                   (Water 3))
           (OUTPUTS (Population 1)
                    (MetallicAlloys 1)
                    (MetallicAlloysWaste 1)
                    (Water 2)))
```

Electronics Template

```
(TRANSFORM C
           (INPUTS (Population 1)
                   (MetallicElements 3)
                   (MetallicAlloys 2)
                   (PotentialEnergyUsable 3)
                   (Water 3))
           (OUTPUTS (Population 1)
                    (Electronics 2)
                    (ElectronicsWaste 1)
                    (Water 2)))
```

Transformation templates restrict the kinds of transformations that can take place (e.g., a country cannot arbitrarily transform a combination of "coal" and "water" into "gold"), so we have to agree on templates in advance. Some product outputs from a transform will correspond to "waste" material, which is also represented as a resource, albeit a typically undesirable one.

Our assumption is that a TRANSFORM template applies to any country (i.e., any country can be bound to variable C) that has the requisite inputs.

The non-divisible amounts for each resource (e.g., 1 unit of AvailableLand, 1 unit of Housing) can be multiplied to increase the TRANSFORM yields. So if 5 units of Housing are desired, every resource in the HousingTransform template is multiplied by 5 to achieve the desired operation, thus becoming:

```
(TRANSFORM C
           (INPUTS (AvailableLand 5)
                   (Population 25)
                   (Water 25)
                   (MetallicElements 5)
                   (Timber 25)
                   (MetallicAlloys 15)
                   (PotentialEnergyUsable 25))
           (OUTPUTS (Housing 5)
                    (HousingWaste 5)
                    (Population 25)
                    (Water 20)))
```

Transfers

TRANSFER operations (a.k.a. Transfers) represent how a country shares its resources with other countries. Each TRANSFER operation is defined by the following template:

$$\texttt{(TRANSFER C}_i\ \texttt{C}_k\ \texttt{((R}_{j1}\ \texttt{X}_{j1}\texttt{)}\ \texttt{...}\ \texttt{(R}_{jm}\ \texttt{X}_{jm}\texttt{)))}$$

where C_i gives various amount (X) of resource (R) to country C_k. These amounts are subtracted and added from the respective stores of resources in C_i and C_k. The TRANSFER of a list of resources is roughly shorthand for a list of singleton resource transfers, as shown below:

$$\texttt{(TRANSFER C}_i\ \texttt{C}_k\ \texttt{((R}_{j1}\ \texttt{X}_{j1}\texttt{)))}$$
$$\texttt{...}$$
$$\texttt{(TRANSFER C}_i\ \texttt{C}_k\ \texttt{((R}_{jm}\ \texttt{X}_{jm}\texttt{)))}$$

For example, the following operation transfers 3 Housing units from country C_1 to C_2:

$$\texttt{(TRANSFER C}_1\ \texttt{C}_2\ \texttt{((Housing 3)))}$$

Of course, a composite transfer could also be written as a collection of subset transfers, singleton or otherwise. In principle, the composite transfer is informationally equivalent to a collection of subset transfers, but we can imagine that the subset transfers may be interspersed with transfers between other countries, much like transactions can be interspersed in a database management system, and these interspersed transfers may lead to different results in a simulation. ***For Part 1, use only singleton TRANSFER operations*** in schedules that your AI agent explores during search. **Composite TRANSFER operations can potentially be used in Part 2** to implement macro operators (more later).

Resources and Allowable Transformations

Resources are of various types. Some resources are "basic" or "raw" resources, and some resources are "created" (e.g., manufactured) from other resources using the TRANSFORM operator. Some resources have explicit positive or negative worth, represented as positive or negative numeric weights, while other resources only have worth in what they enable in the way of further resource creation or other meta-operations. Every resource created with a positive weight has an associated resource with a negative weight, representing waste from the creation process. Negatively-valued resources have the effect of counter-balancing the world state.

In principle, some resources are renewable, and some are not. An omnipotent "simulation manager" might control the auto-incrementing (renewal) and auto-decrementing (natural loss) of some such resources. **In Part 1 we will consider all resources to be non-renewable, but you can model renewal in Part 2.** Additionally, some raw resources can be partially reclaimed from created resources through recycling. Reclamation could be done through the TRANSFORM operator, and there may be explicit waste associated with it, but "waste" also can be implicit in that some of the original raw resources are lost. **Recycling can also be modeled in Part 2 of the project.**

As noted earlier, transformation templates constrain the kind of resource transformations that can take place. Most kinds of resources, positive or negative, can be transferred between countries, but some can't (e.g., solar power, land). Raw resources might include:

- Population (renewable)
- Metallic Elements (non-renewable)
- Timber (renewable)
- Available Land (non-renewable)
- Potential Renewable Energy (solar, wind, water)
- Potential Non-Renewable Fossil Energy (oil, coal, natural gas)
- Water (renewable)

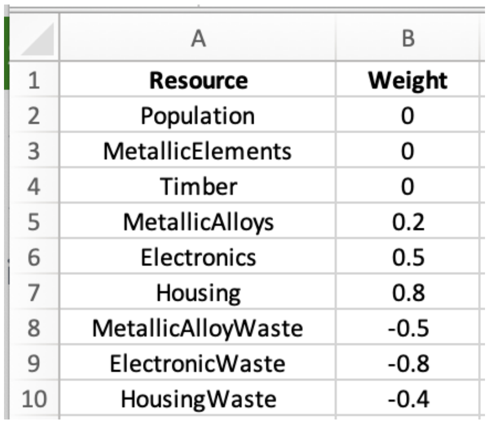
Created Resources might include:

- Military and Military Waste
- Metallic Alloys and Metallic Alloy Waste
- Housing and Housing Waste
- Food and Food Waste
- Potential Fossil Energy Usable (such as extracted and refined fossil fuels) and Potential Fossil Energy Usable Waste (e.g., waste as a result of the extraction process)
- Potential Renewable Energy Usable (solar panels, dams, windmills) and Potential Renewable Energy Usable Waste
- Electronics and Electronics Waste

As noted earlier, to execute a TRANSFORM, a country must have adequate input resources. For a country to TRANSFER resources to another country, the source country must have all the resources in hand. These requirements for the availability of adequate resources are called ***preconditions*** of the TRANSFORM and TRANSFER operations.

Representing a Virtual World

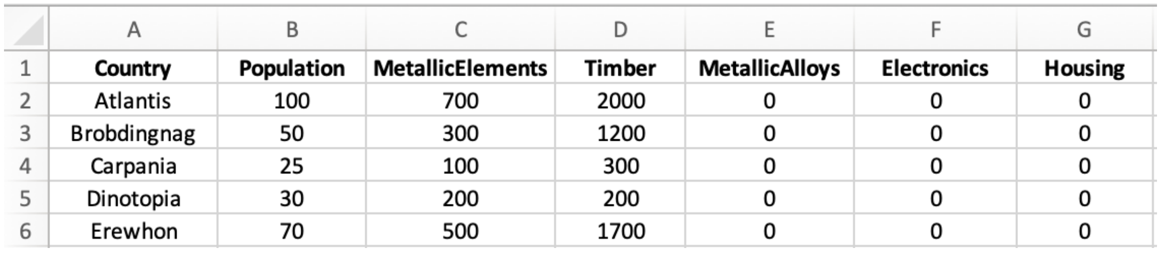
A virtual world state is a set of virtual countries, each containing some amount of various resources. You may or may not choose to weight these resources, as shown below:



	A	B
1	**Resource**	**Weight**
2	Population	0
3	MetallicElements	0
4	Timber	0
5	MetallicAlloys	0.2
6	Electronics	0.5
7	Housing	0.8
8	MetallicAlloyWaste	-0.5
9	ElectronicWaste	-0.8
10	HousingWaste	-0.4

Figure 1: Sample Resources and Weights

An initial world state might include the following:



	A	B	C	D	E	F	G
1	**Country**	**Population**	**MetallicElements**	**Timber**	**MetallicAlloys**	**Electronics**	**Housing**
2	Atlantis	100	700	2000	0	0	0
3	Brobdingnag	50	300	1200	0	0	0
4	Carpania	25	100	300	0	0	0
5	Dinotopia	30	200	200	0	0	0
6	Erewhon	70	500	1700	0	0	0

Figure 2: Sample Initial World State

Implicit in this illustration is that one set of resource weights applies to all virtual countries. We will make this assumption in Part 1, but you might relax this assumption in Part 2, reflecting the idea that different nations value resources differently.

Sequences of Operations

Transfers and transformations will typically be done in sequence, as in the case of trades:

```
(TRANSFER C_i C_k ((Timber 25)))
(TRANSFORM C_k (INPUTS (Timber 50)..) (OUTPUTS (Housing 10)))
(TRANSFER C_k C_i ((Housing 5)))
```

This example indicates that C_i gives C_k Timber, C_k transforms Timber and other materials into Housing, and then C_k gives some Housing back to C_i. In this example, a package might omit the middle TRANSFORM, being agnostic about how C_k acquired the Housing to trade back (e.g., through manufacture, trade, or already in possession).

Your AI agent in Parts 1 and 2 will correspond to one country, call it 'self' for now, and sequences will be found by your AI that presumably benefit 'self'. For example:

```
(TRANSFER C_i self ((Timber 25)))
(TRANSFORM self (INPUTS (Timber 50)..) (OUTPUTS (Housing 10)))
(TRANSFER self C_i ((Housing 5)))
```

In a fuller simulation, perhaps in Part 2, your AI might discover a sequence for 'self' like the one above and then pitch it as a candidate to other countries, C_i, to see if one might agree to the trade.

Resources that were traded back and forth would typically be disjoint, but there is nothing about the definition of TRANSFER that precludes resources being given, and then given back in whole or in part. Even "waste" resources can be transferred, presumably with deleterious effects on the recipient (but in the real world a country can pay another country to offload its waste).

Sequences can be arbitrarily long, implementing more sophisticated interactions:

```
(TRANSFER C_i C_k ((...)))
(TRANSFER C_k C_p ((...)))
(TRANSFER C_p C_i ((...)))
(TRANSFER C_p C_k ((...)))
```

In this example, C_i gives resources to C_k, which then transfers resources to C_p, which then distributes resources back to C_i and C_k. There may be TRANSFORMS or other trades included as well.

Project Part 1

Programming Project Part 1 requires you to complete an anytime, forward-searching, depth-bounded, utility-driven scheduler. The goal is to create schedules for a particular country, but the state of the world will impact the state of each country in at least one way.

The top level function is prototyped as:

```
def country_scheduler(your_country_name, resources_filename,
                      initial_state_filename, output_schedule_filename,
                      num_output_schedules, depth_bound,
                      frontier_max_size)
```

where most of the parameters are explained in the section **Input and Output**, but the final two parameters are described in the section **Search Strategy**. The exact format of this function will depend of the programming language that you use.

Input and Output

Each implementation will read the following information from a CSV file:

- **resources_filename:** Names, weights, and factor definitions of the resources (refer to the next section on *The State Quality of a Country*). See Figure 1 above.
- **initial_state_filename:** The initial state of the world in terms of resource amounts per country (this is also where you define the country labels). See Figure 2 above.

You can post to the Discussion Forum the code for translating input files to internal representations. This could positively impact your community score because it has opportunities for citation credit. This is the only code you can post without prior permission.

Each implementation will write out an ordered list of **num_output_schedules** to **output_schedule_filename** for each initial state problem. The expected format of the output schedules is specified for you below.

The *State Quality* of a Country

The *state quality* of a country is determined by each student, though you can share quality functions on the Discussion Forum and in-class in a form that is comprehensible to others, but not in the form of code. Your state quality function, whatever it is, must be substantively dependent on a country's resources. For example, one state quality function could be a weighted sum of resource factors, and it could be normalized by the population resource, such as $w_{Ri} * c_{Ri} * A_{Ri}/A_{Population}$, where A_{Ri} is the amount of a resource, and c_{Ri} is a proportionality constant (e.g., 2 units food per person, 0.5 houses per person). Or the quality function could be an ecological footprint that is normalized by the AvailableLand resource. State Quality could take other forms as well.

In any case, some kind of weighting of resources will likely be important. Your choice of state quality function should be informed by one or more sources of what are relevant measures of country health in the real world (though avoid getting "into the weeds"), and you can share these sources over the Discussion Forum or live in class. Generally, I expect and hope that you will share your ideas on state quality freely on the Programming Project Discussion Forum.

Resources

For Part 1 of this project, the following exhaustive list of resources may be used. Note that resources marked with a * indicate resources that **must** be included when defining states, computing state quality, and the like:

- AvailableLand
- Water
- **Population***
- PopulationWaste
- **MetallicElements***

- **Timber***
- Farm (Farm is obtained from AvailableLand and reduces the AvalilableLand amount)
- FarmWaste
- **MetallicAlloys***
- **MetallicAlloysWaste***
- **Electronics***
- **ElectronicsWaste***
- **Housing***
- **HousingWaste***
- Food (Food is enabled by the Farm resource, perhaps accompanied by Water)
- FoodWaste
- PotentialFossilEnergy (e.g., oil)
- PotentialFossilEnergyUsable (e.g., oil that is extracted from land, can be exported)
- PotentialFossilEnergyUsableWaste (e.g., waste as a result of the extraction process)
- PotentialRenewableEnergy
- PotentialRenewableEnergyUsable
- PotentialRenewableEnergyUsableWaste

For the required (*) created resources, you may either use the transformation templates given in the *Project Background* section, or you may use the following transformations which only utilize required resources. Note that you can alter these templates in your AI agent, so long as you cite your sources to justify the alteration:

```
Housing Template

(TRANSFORM C
           (INPUTS (Population 5)
                   (MetallicElements 1)
                   (Timber 5)
                   (MetallicAlloys 3))
           (OUTPUTS (Housing 1)
                    (HousingWaste 1)
                    (Population 5)))

Alloys Template

(TRANSFORM C
           (INPUTS (Population 1)
                   (MetallicElements 2))
           (OUTPUTS (Population 1)
                    (MetallicAlloys 1)
                    (MetallicAlloysWaste 1)))
```

Electronics Template

```
(TRANSFORM C
           (INPUTS (Population 1)
                   (MetallicElements 3)
                   (MetallicAlloys 2))
           (OUTPUTS (Population 1)
                    (Electronics 2)
                    (ElectronicsWaste 1)))
```

Including one or more other resources can get you additional points, particularly if you have sources to support, even vaguely, your design of additional TRANSFORM operations. Every resource that has an associated Waste resource, must be accompanied by that Waste resource in your state definition. FYI: Water is involved (in the real world) in virtually every transformation (e.g., https://www.patagonia.com/our-footprint/organic-cotton.html), and the inclusion of water would be an example of citing sources to support your transformation definitions.

The Format of a Schedule

A schedule is formatted as a sequence (list) of TRANSFORM and TRANSFER operations that are each fully grounded (i.e., all variables are replaced by constants). For example,

```
[ (TRANSFORM C1
             (INPUTS (Population 25)
                     (MetallicElements 5)
                     (Timber 25)
                     (MetallicAlloys 15))
             (OUTPUTS (Housing 5)
                      (HousingWaste 5)
                      (Population 25)))
  (TRANSFER C1 C2 ((Housing 3)))
  ...
]
```

though probably with C1 and C2 bound to particular countries, one of which is typically (but not necessarily) your country. (An aside: allowing variables in a schedule that represent another country is often beneficial and allows for flexibility at schedule execution time, but I would ground all variables to constants in your submitted schedules).

The first operator in the schedule will change the initial state into a second state *with certainty*, the second operator in the schedule will change the second state into a third state *with certainty*, etc. Thus, there are *no uncertainties associated with the effects of operators*. The state that precedes an operator's application must satisfy the preconditions of that operator (i.e., the country that is executing a transform or transfer must have sufficient resources). A schedule in which each operator's preconditions are satisfied just prior to the operator's application is called a legal schedule.

Undiscounted Reward of a Schedule

The reward of a schedule can be positive or negative, and is the difference between the state quality of the end state of the schedule for a country and the state quality of the start state for the same country.

A schedule can benefit or degrade countries to varying extents, so each country that participates in a schedule probably has a different reward for a given schedule.

Remember, you design how to implement state quality for a country as discussed above, but you must use the difference between the beginning and end state qualities as the *undiscounted reward*:

$R(c_i, s_j) = Q_{end}(c_i, s_j) - Q_{start}(c_i, s_j)$, for country c_i and schedule s_j.

Discounted Reward of a Schedule

The Discounted Reward comes from the end state of a schedule, but the farther the end state is into the future, the less it counts for a country. If there are N time steps in a schedule, then the discounted expected reward for a country is:

$DR(c_i, s_j) = gamma^N * (Q_{end}(c_i, s_j) - Q_{start}(c_i, s_j))$, where $0 <= gamma < 1$.

For many sequential decision problems (Chapter 17, Russell and Norvig; Chapter 9, Poole and Mackworth), each state in the sequence comes with some reward (positive or negative/penalty), and the utility of the sequence is the sum of discounted state rewards, but in Part 1, we assume that the entire reward comes at the end state only. Nonetheless, your system can (and probably will) compute the discounted end state quality for every partial schedule on the search frontier, which could be used to organize the frontier as a priority queue (refer to the *Expected Utility of a Schedule* section). You will experiment with different values of gamma and can report results on the Discussion Forum.

Probability that a Schedule will Succeed

Even when there is no uncertainty associated with an operator's effects when the operator is applied, and therefore no uncertainty associated with a legal schedule's effects when the schedule is applied, there remain other sources of uncertainty in the schedule. Notably, other countries referenced in the schedule may not "go along" with the schedule if an attempt were made to "execute" it in the real world. You will use state qualities for other countries as well as your own country, to judge the likelihood that a schedule will be accepted by all parties. Notably, if a schedule references a number of other countries, then the more that the schedule benefits each of the referenced countries (i.e., in terms of discounted reward), the chances that they will agree to the schedule increase.

The probability that a country, c_i, will participate in a schedule, s_j, is computed by the logistic function where x corresponds to $DR(c_i, s_j)$ and $L = 1$, and you can experiment with different values of x_0 and k (but use $x_0 = 0$ and $k = 1$ as starting points). Think about, and report on, how different parameter settings might reflect biases in the real world (e.g., a reason for shifting x_0 might be to reflect opportunity costs — what other benefits might await a patient country?).

Given the individual probabilities of each referenced country participating, $P(c_i, s_j)$, the probability that a schedule will be accepted and succeed, $P(s_j)$, could be computed in a number of ways (e.g., the min of the probabilities, reflecting the weakest link), but we'll use the product of the probabilities of the individual $P(c_i, s_j)$.

Note that the strategy for estimating the probability that a schedule is accepted (i.e., will be accepted by all parties to the schedule) and succeeds, does not come from statistics accumulated over data from the "real world" (to include game play) as we might think is ideal, but our method draws from an information theoretic tradition of estimating probabilities of "events" from the "descriptions" of those events. An assumption in this description-driven methodology is a bias that more "complicated" descriptions represent lower probability events. It is a quantification of Occam's razor.

This is the one way (probably the only way given the time constraints in Part 1) that your scheduler for your country will take into account the state qualities of other countries, as well as your own. It has the effect of countering ill-considered greed.

Expected Utility of a Schedule

The probability that a schedule will be accepted and succeed (which takes into account other countries) multiplied by the discounted reward for your country (self) is the central factor in computing the expected utility of a schedule (s_j) for your country (self, c_i). But what is the cost to a country of producing a schedule that would ultimately fail? For simplicity, let the cost of the failure case be a negative constant, C. You choose C, with justification given, or if you are more ambitious, design a more general function to represent the failure cost:

$EU(c_i, s_j) = (P(s_j) * DR(c_i, s_j)) + ((1 - P(s_j)) * C)$, where $c_i = self$

Summarizing the Interdependence of Measures

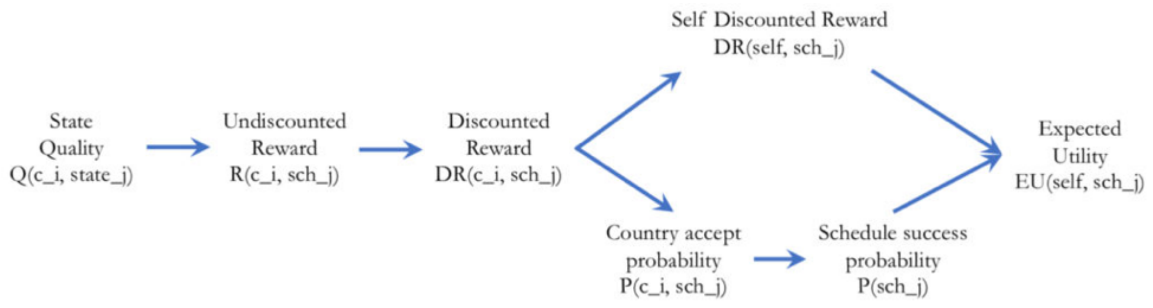

State Quality Q(c_i, state_j)
Undiscounted Reward R(c_i, sch_j)
Discounted Reward DR(c_i, sch_j)
Self Discounted Reward DR(self, sch_j)
Country accept probability P(c_i, sch_j)
Schedule success probability P(sch_j)
Expected Utility EU(self, sch_j)

Figure 3: Interdependence Between Measures

Search Strategy

Your search strategy is expected to be an anytime, forward-searching, depth-bounded, utility-driven scheduler. The search is *country centric*, and at least one country variable, C, in any final schedule should be bound to 'self'. It may well be the case that your search implementation insists that every operator in a schedule has a country variable bound to 'self'. Each student decides on this constraint.

By *forward searching*, each problem starts with an initial world state, and searches into the future. This is NOT a regression (or backward) search. As a suggestion, use the skeletal algorithms for generic search discussed during the live lectures or in one of the textbooks (Russell and Norvig, Poole and Mackworth), which use an explicit frontier data structure. In most search strategies, the frontier will be a priority queue, but in the case of (heuristic) depth-first search, the frontier will be a stack.

A *utility-driven scheduler* means that a utility measure organizes the priority queue that we are calling the *frontier*. Use $EU(self, s_j^{'})$ to organize the search frontier, where $s_j^{'}$ is a partial schedule on the frontier that is at a depth less than the depth-bound.

The *depth-bound* specifies the maximum depth to search (assuming the initial state is at depth 0); in other words, the maximum depth of nodes on the frontier. It also defines the depth that your implementation must search along every search path, except in a rare situation when a path cannot be advanced further (i.e., the *generate successors* function yields the empty set for a state on the frontier). Thus, you can't just search to depth 1 by a single execution of the generate-successors function to the initial state). Importantly, for each state of each final schedule (path) that is found by your scheduler, the $EU(self, s_j^{'})$ value of the partial schedule, $s_j^{'}$, should accompany the schedule, which will also include the $EU(self, s_j)$ of the entire schedule, s_j. Thus, schedules in the output file **output_schedule_filename** will be formatted like:

```
[ (TRANSFORM self ...) EU: S_1
  (TRANSFER self C2 ((Housing 3))) EU: S_2
  ...
  (TRANSFORM self ...) EU: S_N
]
```

The reason for printing out these intermediate values is to see whether frontier scores can get worse before they get better, as well as to easily identify the best sub-schedule within a schedule that goes to the depth bound.

Priority queue size will limit the size of the frontier and thus the extent of search, but you will choose what to place in the queue. Will you simply place the top scoring partial schedules found throughout the search (even if they share almost all ancestors), or will you place schedules found through different paths to implement a search with greater ancestral diversity?

Finally, your *anytime implementation* will continue to search for schedules at depth-bound even after the first such schedule is found. When adapting the generic search algorithm in your intro texts, remember that these algorithms are goal based, and there are no goals per se in your implementation, or alternatively you will consider schedules at depth-bound as "goals", and keeping track of the best schedule in terms of EU found so far will complete your anytime implementation. In terms of your final report, for each problem, you will report the EU scores in the order that they were discovered, and create a scatter plot of these to see if better schedules tend to be discovered early, as we would hope in an anytime search.

Part 1 Deliverables

The primary narrative of Part 1 of the Programming Project will be in the form of Powerpoint slides and a video presentation using those slides. You will submit a zip file (see the Syllabus for the due date) containing the following files as your only deliverable (the names of the files should be exactly as outlined in **bold**):

- **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 10 minutes). The slides and video will focus on those aspects of the project that were *of your design* and not pre-specified for you, but your video should still be self-contained; therefore, don't omit discussing pre-specified components. Simply use them as connective material to tell the whole story focused on your contributions:
    - Give an intro to the "big picture" of the project as you understand it
    - Describe your State Quality function and a justification of it (which can be intuitive, but sources can be cited here too and would be desirable)
        - Summarize the pipeline of scores being used, from state quality, through rewards and schedule success probabilities, to expected utility (you can reuse the image you were given and/or create your own)
        - Include your thoughts, possibly informed by experiments, on certain parameter settings (e.g., why you chose certain values for x_0, k, gamma, etc.)
    - Describe how you specialized the modified generic search function
        - State how the frontier is organized (a priority queue by EU or something else, and something other than a strict priority queue)
        - Characterize the forward, utility-driven, depth-bounded search strategy that you converged on:
            - Best-first search?
            - Coupled with beam search?
            - Heuristic depth-first search?
            - Iterative deepening heuristic depth-first search?

- Give two or more test case(s) that you regard as interesting and informative. Each test case minimally includes:
  - An initial state,
    - Why this state? Does it illustrate world balance or certain imbalance(s) that might encourage or discourage cooperation?
  - Parameter settings
  - Three selected output schedules for each case, and the EU scores for each state in each schedule
    - Observations about these schedules, perhaps the relative frequency of transfers and transforms
  - Interesting observations, both about the example test cases, and perhaps trends across all test cases, such as whether EU is monotonically increasing or decreasing, or neither, over example test cases and other schedules that are not shown in detail
- Additional topics to include in your presentation might be:
  - Whether you used hand-crafted macro operators, examples of them, and your assessment of whether they benefited the search or not
  - Summaries of experimental results, perhaps in graphical or tabular forms, such as:
    - **How runtime and/or the EU of schedules varied with different parameter settings (e.g., depth-bound, beam size). If you know/find how to instrument code in your language of choice to determine runtime, please share with the group. This is more or less required.**
    - A scatter graph on the EU (y-axis) of final schedules, with an x-axis that is the order in which the schedules were written to the output file (in an anytime planner, it is better if good schedules are found earlier than poor schedules)
    - I am more concerned that you do meaningful tests than with how your simulator actually performs on the tests
    - A project without an evaluation, typically quantitative but possibly qualitative, won't get a maximum score
  - Transform templates that you added to the set that was given to you, and citations to any sources
  - Interesting observations about connections to the real world that were not addressed earlier (ideally with citations)

- Citations to outside sources that you used (e.g., on transforms and resources, algorithmic variations), **including references to other students, which will affect your 'community score' and theirs (a source of extra credit)**. You can give citations to other students throughout, but summarize the citations on a single slide at the end of the presentation slides.

- **SourceCode.zip:** Well-documented and well-formatted code, including:
  - **README:** Containing directions for running your code and identifying the file containing the `country_scheduler` function, and any other important top-level functions (e.g., one that iterates through parameter settings that you are experimenting with)
  - **input.zip:** your actual INPUT FILES, including a file for each test case that you illustrated in the video, with appropriate indenting
  - **output.zip:** your OUTPUT FILES, with clear correspondence with input files based on naming conventions, as appropriate
  - **test_cases_summary.pdf:** A PDF containing a nicely formatted and documented list of test cases results. This file will be informationally redundant with other files that you are also submitting, but the emphasis here is on ONE easily comprehensible source in which we can look at a summary of results. This file is the closest thing to a written report that you will submit, so include some connective, explanatory text, and motivate and explain why you designed each test case. Test cases should include the examples that you used in your presentation, and each test case must include:
    - The initial state
    - Various parameter settings:
      - `your_country_name`
      - `resources_filename`
      - `initial_state_filename`
      - `output_schedule_filename`
      - `num_output_schedules`
      - `depth_bound`
      - `frontier_max_size`
      - x_0,
      - k,
      - $gamma$,
      - failure cost, C,
      - any constants involved in your state quality function
    - A list of at least 5 output schedules (and intermediate and final state scores)