AI Final Project Part 1

Brian Goldsmith

Anytime, Forward-Searching, Depth-Bounded, Utility-Driven Scheduler ... that works???

- Search Algorithm
- State Quality Function
- Resource Weights

The primary goal of the project was to gain a better understanding of heuristic search algorithms and expected utility by designing an AI agent to choose “good” actions in a world trade game. While the search algorithm was a portion of the project, my attention and focus was primarily spent on the state quality function and the weights since that determined the Expected Utility.

State Quality Function

Housing - The average household in the United States is 2.5 people[1] which matches most of North America and Europe.[2] The highest average household size is in Senegal with 8.33 people per house.[3]

Electronics - Currently, people in North America have roughly 13.4 devices and connections, while the Middle East and Africa have 1.5. The current world-wide average is roughly 2.4 devices and connections.[4]

1. https://population.un.org/Household/index.html#/countries/
2. https://en.wikipedia.org/wiki/List_of_countries_by_number_of_households
3. https://ceoworld.biz/2020/02/19/these-are-the-countries-with-the-largest-household-size/
4. https://www.statista.com/statistics/1190270/number-of-devices-and-connections-per-person-worldwide/

When determining the weight for the housing resource, I thought it would be useful to look at average household sizes of different countries. Interestingly, the United States averages roughly 2.5 people per household. Typically, North American and European countries have lower averages, however many African, Asian, and Middle Eastern countries have much higher averages. Including Senegal with the highest average of 8.33. Initially, I set three different goals for housing. First, the country should try to prioritize 8 people per house. Once that has been accomplished then in future housing decisions should be less of a priority. Again, once housing has reached 4 people per house then again housing should be even less of a priority. Finally, when 2.5 people per house is reached, then housing should be considered done and should be given the lowest priority.

I took a similar approach with electronics. Currently, people in North America have roughly 13.4 devices and connections, while the Middle East and Africa have 1.5. The current average is roughly 2.4 devices and connections. While electronics are a great sign of technological progress, when starting the search other resources (housing, food, etc) should have higher priorities. However, as those other resources are being met, electronics should rise in priority. The first goal for electronics will be 1 electronics for 8 people to represent 1 electronic per household (given the initial housing goal.) After this tier is reached, then I plan to decouple the weight from housing and aim for 1 electronics per person. After 1 electronics per person, I will

change the priority again until the 10 electronics per person is reached. Once that tier has been met, then a single constant priority weight will be given.

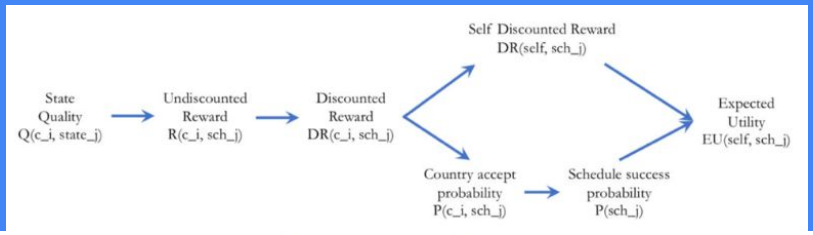
State Quality Function and Resources

$$\sum \frac{resource_i}{population} \; x \; weight_i$$

Resource	Weight	Factor
Population	0	0
MetallicElements	0	1
Timber	0	1
MetallicAlloys	0.2	1
MetallicAlloysWaste	-0.05	1
Electronics	0.7;0.25;0.15	0.125;1;10
ElectronicsWaste	-0.8	1
Housing	1;0.6;0.0	0.125;0.25;0.4
HousingWaste	-0.4	1

I used a simple state quality function of summing the resource/population * weight. By using a factor with the housing and electronics resource, I would adjust the weight if the ratio of a resource to a population was at a certain tier. As we will see later, this turned out to be more complicated than expected.

Expected Utility



- Undiscounted Reward: $R(c_i, s_j) = Q_{end}(c_i, s_j) - Q_{start}(c_i, s_j)$
- Discounted Reward: $DR(c_i, s_j) = \gamma^N * R(c_i, s_j)$ where: N = num time steps
- Country Accept Probability: $P(c_i, s_j) = {}^{1}/_{1+e^{-k(DR(c_i,s_j)-x_0)}}$ where: x_0 is the sigmoid midpoint; k affects curve steepness
- Schedule Success Probability: $P(s_j) = \prod_{c \in C} P(c, s_j)$
- Expected Utility: $EU(self, s_j) = P(s_j) * DR(self, s_j) + \left(\left(1 - P(s_j)\right) * C\right)$

The expected utility function along with the other equations were provided and I would prefer not to dwell on them on this slide. I will address some things specific to the expected utility pipeline when discussing some of the results.

Search Function

Priority Queue using the Expected Utility

Pop the node with the largest expected utility, if at the max depth then add the schedule to the list of final schedules, otherwise generate successors.

First go through all possible transforms, then all transfers adding to the queue

My search function was constructed form the generic graph searching algorithm presented in the async lecture utilizing the expected utility as a heuristic. I re-used the sample priority queue then wrote the code to generate the successors. The search algorithm pops the node with the largest expected utility from the priority queue and then checks to see if the max depth has been reached. If so, the schedule added to the list of final schedules and the search algorithm continues unless the maximum number of schedules has been reached. If not, the search algorithm expands the successors starting with transforms first and then doing transfers, adding new nodes to the priority queue when viable.

Initial State

	Country	Population	MetallicElements	Timber
1	Atlantis	331	48	501
2	Brobdingnag	399	350	399
3	Carpania	145	99	362
4	Dinotopia	212	480	351
5	Erewhon	37	54	301

https://www.mapsofworld.com/world-top-ten/countries-with-most-timber-producing-countries.html
https://en.wikipedia.org/wiki/List_of_countries_by_iron_ore_production
https://www.worldometers.info/world-population/population-by-country/

For the initial state, I reviewed the listed sites with Atlantis using the stats for the United States, Brobdingnag using China's stats, Carpania using Russia's stats, Dinotopia using Brazil's stats, and Erewhon using Canada's stats.

Tests and Results

num_output_schedules = 3, depth_bound = 5, frontier_max_size = 10,000

```
[ (TRANSFORM self (INPUTS (Population 3)(MetallicElements 6)) (OUTPUTS (Population 3)(MetallicAlloys 3)(MetallicAlloysWaste 3)) EU: 0.0006797583081570998
(TRANSFORM self (INPUTS (Population 3)(MetallicElements 6)) (OUTPUTS (Population 3)(MetallicAlloys 3)(MetallicAlloysWaste 3)) EU: 0.0006797583081570998
(TRANSFORM self (INPUTS (Population 5)(MetallicElements 10)) (OUTPUTS (Population 5)(MetallicAlloys 5)(MetallicAlloysWaste 5)) EU: 0.0006231117824773415
(TRANSFORM self (INPUTS (Population 2)(MetallicElements 4)) (OUTPUTS (Population 2)(MetallicAlloys 2)(MetallicAlloysWaste 2)) EU: 0.000368202416918429
(TRANSFORM self (INPUTS (Population 11)(MetallicElements 22)) (OUTPUTS (Population 11)(MetallicAlloys 11)(MetallicAlloysWaste 11)) EU: 0.0003398791540785499
]
[ (TRANSFORM self (INPUTS (Population 3)(MetallicElements 6)) (OUTPUTS (Population 3)(MetallicAlloys 3)(MetallicAlloysWaste 3)) EU: 0.0006797583081570998
(TRANSFORM self (INPUTS (Population 4)(MetallicElements 8)) (OUTPUTS (Population 4)(MetallicAlloys 4)(MetallicAlloysWaste 4)) EU: 0.0007930513595166163
(TRANSFORM self (INPUTS (Population 4)(MetallicElements 8)) (OUTPUTS (Population 4)(MetallicAlloys 4)(MetallicAlloysWaste 4)) EU: 0.0006231117824773415
(TRANSFORM self (INPUTS (Population 2)(MetallicElements 4)) (OUTPUTS (Population 2)(MetallicAlloys 2)(MetallicAlloysWaste 2)) EU: 0.000368202416918429
(TRANSFORM self (INPUTS (Population 11)(MetallicElements 22)) (OUTPUTS (Population 11)(MetallicAlloys 11)(MetallicAlloysWaste 11)) EU: 0.0003398791540785499
]
[ (TRANSFORM self (INPUTS (Population 3)(MetallicElements 6)) (OUTPUTS (Population 3)(MetallicAlloys 3)(MetallicAlloysWaste 3)) EU: 0.0006797583081570998
(TRANSFORM self (INPUTS (Population 5)(MetallicElements 10)) (OUTPUTS (Population 5)(MetallicAlloys 5)(MetallicAlloysWaste 5)) EU: 0.0009063444108761329
(TRANSFORM self (INPUTS (Population 3)(MetallicElements 6)) (OUTPUTS (Population 3)(MetallicAlloys 3)(MetallicAlloysWaste 3)) EU: 0.0006231117824773415
(TRANSFORM self (INPUTS (Population 2)(MetallicElements 4)) (OUTPUTS (Population 2)(MetallicAlloys 2)(MetallicAlloysWaste 2)) EU: 0.000368202416918429
(TRANSFORM self (INPUTS (Population 11)(MetallicElements 22)) (OUTPUTS (Population 11)(MetallicAlloys 11)(MetallicAlloysWaste 11)) EU: 0.0003398791540785499
]
```

The results of the test above were a bit confusing for a few reasons. The search algorithm was finding getting to the same last two states in all three schedules. Also, a major concern is that the EU values are VERY small. Since all schedules use the same formulas, it may seem like this shouldn't be a problem as the values are only important relative to each other. However, the use of the discounted reward in the formulas is critical, especially when dealing with transfers and the probability of a country accepting a transfer.

Tests and Results

k=1
x_0=0
DR is very small
C= -0.5

Country Accept Probability: $P(c_i, s_j) = {}^{1}/_{1+e^{-k(DR(c_i,s_j)-x_0)}}$

Schedule Success Probability: $P(s_j) = \prod_{c \in C} P(c, s_j)$

$-k(DR(c_i,s_j)-x_0$ is then -1(very small - 0)

-1(very small) is going to be close to 0
$e^0=1$

$P(c_i,s_j) \simeq 1/2$ for all countries

$P(s_j) \simeq 1/2 * 1/2 \simeq 1/4 \simeq 0.25$

$EU = P(s_j)*DR(self, sch\_j)+((1-P(s_j))*C)$

$EU \simeq 0.25 *$ very small $+ ((1-0.25)*-0.5)$

$EU \simeq 0.25 *$ very small -0.375

EU is ALWAYS negative

When looking at the math used by the country accept probability, schedule success probability, and expected utility, it can be seen that the country accept probability is always very close to 1/2. Since the schedule success probability is the product of the two countries involved in the transfer, 1/2 * 1/2 will be roughly 1/4. When used in the expected utility function it is very clear that the EU will always be negative for transfers which is a pretty significant issue.

Adjustment

Multiply the state quality function by 1000

Adjust weights for Metallic Elements and Timber (both set to 0.05)

Lowering C (negative constant) to -0.25

To try to correct the issue with the very small state quality, undiscounted reward, and discounted reward values, I multiplied my previous state function by 1000 since the final EUs from the first run were mostly 3 to 4 places behind the decimal. I found that since countries only had natural resources the weight that was assigned to zero the result of the state quality function was still zero. I therefore modified the weights of the natural resources with a positive weight to help the state quality values. Also, in order to try to help transfers, I lowered the negative constant C which is used in the expected utility function to -0.25.

Test and Results

```
[ (TRANSFORM self (INPUTS (Population 22)(MetallicElements 44)) (OUTPUTS (Population 22)(MetallicAlloys 22)(MetallicAlloysWaste 22)) EU: 1.6616314199395745
(TRANSFER Brobdingnag self (MetallicElements 3)) EU: 0.18970755442848553
(TRANSFORM self (INPUTS (Population 1)(MetallicElements 2)) (OUTPUTS (Population 1)(MetallicAlloys 1)(MetallicAlloysWaste 1)) EU: 0.6231117824773396
(TRANSFORM self (INPUTS (Population 1)(MetallicElements 2)) (OUTPUTS (Population 1)(MetallicAlloys 1)(MetallicAlloysWaste 1)) EU: 0.3209969788519631
(TRANSFORM self (INPUTS (Population 5)(MetallicElements 10)) (OUTPUTS (Population 5)(MetallicAlloys 5)(MetallicAlloysWaste 5)) EU: 0.1841012084592144
]
[ (TRANSFORM self (INPUTS (Population 22)(MetallicElements 44)) (OUTPUTS (Population 22)(MetallicAlloys 22)(MetallicAlloysWaste 22)) EU: 1.6616314199395745
(TRANSFER Brobdingnag self (MetallicElements 3)) EU: 0.18970755442848553
(TRANSFORM self (INPUTS (Population 2)(MetallicElements 4)) (OUTPUTS (Population 2)(MetallicAlloys 2)(MetallicAlloysWaste 2)) EU: 0.6419939577039262
(TRANSFORM self (INPUTS (Population 1)(MetallicElements 2)) (OUTPUTS (Population 1)(MetallicAlloys 1)(MetallicAlloysWaste 1)) EU: 0.3304380664652573
(TRANSFORM self (INPUTS (Population 4)(MetallicElements 8)) (OUTPUTS (Population 4)(MetallicAlloys 4)(MetallicAlloysWaste 4)) EU: 0.1841012084592144
]
[ (TRANSFORM self (INPUTS (Population 22)(MetallicElements 44)) (OUTPUTS (Population 22)(MetallicAlloys 22)(MetallicAlloysWaste 22)) EU: 1.6616314199395745
(TRANSFER Brobdingnag self (MetallicElements 3)) EU: 0.18970755442848553
(TRANSFORM self (INPUTS (Population 1)(MetallicElements 2)) (OUTPUTS (Population 1)(MetallicAlloys 1)(MetallicAlloysWaste 1)) EU: 0.6231117824773396
(TRANSFORM self (INPUTS (Population 2)(MetallicElements 4)) (OUTPUTS (Population 2)(MetallicAlloys 2)(MetallicAlloysWaste 2)) EU: 0.3304380664652573
(TRANSFORM self (INPUTS (Population 4)(MetallicElements 8)) (OUTPUTS (Population 4)(MetallicAlloys 4)(MetallicAlloysWaste 4)) EU: 0.1841012084592144
]
```

While the EU is much higher, the lack of variety in the schedules was concerning. While I was happy that a transfer was included, I was surprised that housing and electronics did not appear as both have extremely high weighting. It was at this point that I realized my planned for tiered weighting would not work.

Tiered Weighting Failure

Population: 331

Housing: weight of 1.0 while less than 42 houses
weight of 0.6 when between 42 and 83 houses
weight of 0 after 132 houses

41/331 * 1.0 * 1000 = 123.867
42/331 * 0.6 * 1000 = 76
142/331 * 0.0 * 1000 = 0

Electronics: weight of 0.7 while less than 42 electronics
weight of 0.25 while less than 331 electronics
weight of 0.15 after 3310 electronics

41/331 * 0.7 * 1000 = 86.707
42/331 * 0.25 * 1000 = 31.722
3311/331 * 0.15 * 1000 = 1500.453

I realized that with the tiered weighting, my search algorithm was never going to think that housing and electronics were good choices. My intention was to have the search algorithm focus on housing until there was 1 house per 8 people, then the focus should be on electronics. However, as the amount of housing increases and the weight decreases, the state quality of the country also decreases. When including the resources that are lost and the waste that is gained during the transform process, it never makes sense to choose either. It was at this point that I discovered that what I really wanted to do was split the state quality function so that when deciding a future decision, the weight of housing and electronics should be lower, but when examining the existing states quality it should use the highest value still as more housing and electronics should be a sign of prosperity.

Part 2 Ideas

- Split state quality function to allow for different weights for viewing future decisions versus reviewing the existing resources
- Allies that are more likely to approve transfers
- Each country takes a turn to perform transforms and transfers
- Different search algorithms

While part 1 included many challenges and stumbles, I am looking forward to part 2 where hopefully my work will be focussed more on successes or at least interesting behaviors that are not the result of mistakes. However, due to the less than ideal results, I was forced to step through my search algorithm, as well as the very specific details of the equations making up the expected utility. I would like to try to rethink how to implement tiered weighting, as I find it very practical and fascinating. I believe that since I was unable to run the search at depth levels greater than 6 (which took over 10 minutes), I was not able to really see more variety in the choices made by the search algorithm. Also, it did not help that the other countries were limited to the natural resources since they were not able to perform transforms. Along those lines, I would want to implement a different (and hopefully more efficient) algorithm which should help it get to lower depths.