Final Write-Up

The Problem Space

Our project aims to expand on the Sea of Islands navigation problem that was discussed in the assignment.

The Polynesian Triangle consists of the islands, within Hawaii, Rapanui (Easter Island). Aotearoa (New Zealand). Throughout history canoes have been used to travel between these islands and possibly beyond.

The goal is to plan a network of routes that efficiently distribute items/goods and facilitate travel among the islands in the Sea of Islands. Additionally we aim to identify ways to provide assistance to islands when needed.

We will place emphasis on using representations to depict the Sea of Islands and its relationship with the algorithms discussed in our previous assignment. Each island in our dataset has varying travel times for reaching islands. Not every island has a route to every island making it crucial to optimize travel routes for efficient algorithmic solutions. Also, a directed, weighted graph is used to represent the sea of islands. With this, each island is a node, and the routes with corresponding travel times form the edges.

Furthermore each island possesses characteristics such as population resources, educational resources, native resources and distinct travel experiences. Our primary concern is finding ways to share or circulate types of resources and enable smooth movement of people across the Sea of Islands.

Four Problems/Algorithms

1. Describe an algorithm that looks at the planarity of the Sea of Islands as a graph. It should optimize routes accordingly so that the lines or travel routes connecting them do not overlap or cross as much as possible. Optimizing this helps make the routes between the islands easier to understand and look cleaner so that they are visually clearer. Even if there are many connections, it helps make the map more user-friendly and straightforward to interpret when analyzing the distribution of resources or movement of people. This helps avoid unnecessary intersections or crossings of routes between islands, which gives a more coherent layout to the supplying process.

Kenneth:

Objective

To reiterate, in this directed, weighted graph that represents the Sea of Islands, the nodes are the islands, and the routes are the edges. To start, we must observe and look at the initial layout of

the routes between the islands before they are optimized without thinking about their planarity. Doing this is important since it gives a general idea of how many overlaps or crossings of routes there are in the current layout.

Algorithm Choice

I plan to use a force-directed algorithm, in this case a modified version of Fruchterman-Reingold, since it can help iteratively adjust the positions of both the islands and routes in the Sea of Islands. This allows for finding a layout in which nodes (islands) with their edges (routes) are drawn closer to each other, and the nodes without edges are spaced apart. Although F-R algorithms are mainly used for simple, undirected graphs, I plan to modify this to better suit this case.

It is important to consider the weights of the edges since it influences the layout depending on the travel times in between the islands. Since we have directed edges, directional forces should be added in. This is because it can better show the flow of movement in between the nodes. Also, attractive forces are valuable here. With this, the attractive forces between islands that are connected would pull those nodes closer together, which minimizes the travel times. With this, stronger forces will be used for shorter travel times, and weaker ones will be used for longer travel times. Doing this is convenient and useful to the goal of the algorithm, as it creates layouts that have less edge crossings that are unnecessary.

This F-R algorithm will stop once the number of overlapping edges decreases more slowly, since it will drop significantly in the initial iterations. This means that it will end when the changes in reducing the overlaps become increasingly gradual. It will continue to iteratively adjust the positions of the islands depending on the attractive or repulsive nature of forces between the islands.

Why Fruchterman-Reingold?

I've decided on using the Fruchterman-Reingold algorithm over other options such as Kamada-Kawai. This is because F-R is a better choice at making visually appealing layouts for planar graphs that have less edge crossings. F-R is also effective here since it can also change the length of the edges depending on the number of nodes that are connected to each node. For example, an edge will be longer if it is linked to less nodes, but is shorter when it is connected to more.

This is relevant to the objective of the algorithm since we do not want long routes (edges) that can create unnecessary overlaps or crossings in the graph layout. It is also good for large datasets, especially in the case of the Polynesian Triangle, which has over 1,000 islands. This is important because it shows how it can adapt and adjust for scalability, specifically with the various sets of islands and their connections. Here, the time complexity is $O(|V|^3)$, where V is the

number of vertices or nodes in the graph. Lastly, its space complexity is usually O(|V|), where V is the number of vertices. F-R is better here since it is faster and easier to implement, given that we have a large graph with many nodes.

Why not Kamada-Kawai?
Although K-K is also effective at minimizing edge crossings in planar graphs, it is more complex not only to implement, but also to run. This is important since the graph for the Sea of Islands is made up of over 1,000 nodes, which would make it slow, especially for a large layout. In this case, the algorithm will have worse space complexity compared to F-R which is simpler and efficient. One major drawback is that K-K has parameters that must be adjusted in order to get optimal results. This would need testing and more experimentation, which gives an extra layer of complexity when trying to implement.

Pseudocode

```
function optimizeSeaOfIslands
    initialize positions of islands
    set initial parameters (attractive force, repulsive force)

    repeat until number of overlapping edges decreases slowly
        // attractive forces
        for each edge in seaOfIslands:
            calculate attractive force between connected nodes depending on travel times

        // repulsive forces
        for each pair of islands:
            calculate repulsive force between islands

        // update positions of islands
        for each islands in seaOfIslands:
            calculate net force acting on an island
            update islands' position based on net force

    // return updated version
    return optimized seaOfIslands
```

Conclusion
In short, I will use a modified version of a Fruchterman-Reingold algorithm in order to minimize the number of edge crossings for the graph. This will iteratively look to adjust the node and edge positions for a less-cluttered layout. It will consider travel times as edge weights and the addition of directional forces for better representation of the distribution of resources and people. I've

chosen to use F-R over K-K since it is easier and faster to implement while also being more suitable for larger datasets. Ultimately, doing this makes sure that the final layout is visually clear, and has less visual clutter. This makes it easier for users to visually analyze or interpret the distribution of resources and movement of people between the Sea of Islands.

References

Harel, D., & Koren, Y. (n.d.). *A Fast Multi-Scale Method for Drawing Large Graphs*. Journal of Graph Algorithms and Applications. https://jgaa.info/accepted/2002/HarelKoren2002.6.3.pdf

Kobourov, S. G. (n.d.). *Force-directed Drawing Algorithms*. Brown University. https://cs.brown.edu/people/rtamassi/gdhandbook/chapters/force-directed.pdf

2. (Darrius)

Objective

In the context of the Sea of Islands, this algorithm is responsible for establishing strong and reliable communication towers, because not only is a secure network necessary for everyday internet access, it is extremely crucial for emergency situations. The geography of the islands in relation to each other, but the varying topography of each island poses significant challenges for signal transmission. The algorithm must take all of these factors into account. It will have to account for its signal strength and quality, interference (other technology or highly dense places of internet usage, or natural causes), Island topography (elevation changes, existing structure will cause signal to be less strong), and distance of towers from other towers in-island and out to other islands.

Algorithm Choice

Network Flow Optimization

Main Goal:

The main objective of this algorithm is to create a network that optimizes signal quality and strength, effectively avoiding disruptions as much as possible (or best case scenario).

Algorithm:

The algorithm that would be optimal in this circumstance are network flow algorithms like Maximum Flow and Minimum Cost Flow algorithms. An example of the way this would work is if we have to Islands ‘A’ and ‘B’ and they are, for example, separated by a huge distance across the sea. This algorithm will assess the most ideal path for signal transmission. Using Maximum Flow, it will identify a route that allows the best signal strength, while the Minimum Cost Flow part of the algorithm will make sure the most resource efficient path is chosen. The Ford Fulkerson method for Maximum flow has a complexity of O(max flow * E) which is significant for islands with numerous potential signal paths.

Ford-Fulkerson algorithm for maximum flow problem (2023)

GeeksforGeeks. Available at: https://www.geeksforgeeks.org/ford-fulkerson-algorithm-for-maximum-flow-problem/ (Accessed: 14 December 2023).

Graph Algorithms with Weighted Edges

Create a graph where islands are nodes, and edges will represent possible signal routes. Each tower that is placed will act as a sub node. Then utilize Prim's algorithm to create a minimum spanning tree which will serve our primary goal of optimizing signal coverage. An example of how this algorithm will work is if Island 'C' has a high elevation, this would be taken as a prime place to place a tower, so the algorithm will prioritize connections to this node whereas a place that would be closer to the other tower, because height will precede distance because of how much stronger the signal will be at a higher elevation. The weights of the edges are then calculated based on variables like signal strength, potential interference, and distance. Certain factors such as terrain type, population density, weather patterns, etc… will increase or decrease weight amounts based on whether or not they benefit signal strength. Using a priority queue Prim's algorithm has a time complexity of O(V + E log V) which is efficient for the Sea of Islands and potential tower sites.

Abdelnabi, O. *Minimum spanning tree tutorials & notes:*

Algorithms, *HackerEarth*. Available at: https://www.hackerearth.com/practice/algorithms/graphs/minimum-spanning-tree/tutorial/#:~:text=The%20time%20complexity%20of%20the,priority%20queue%20take%20logarithmic%20time. (Accessed: 13 December 2023).

Tower Placing

Using satellite images to identify areas with the hives potential for signal quality. K- means clustering is optimal for identifying and grouping data points based on proximity. In the context of my problem space, it can analyze user density data to identify areas with high user concentration. Placing towers in or perhaps nearby will ensure that the largest number of users receive strong signal coverage which of course as we know is essential for both everyday connectivity and important communications, and high speed video game connections which again is important for real time gameplay. This algorithm will then cluster several small islands close to each other as a single high demand zone. For example, Island D has a large piece of land without any physical obstructions which is an extremely optimal place to build a signal tower.

The satellite images are then used to analyze user density data, which then identifies this piece of land as a high prio area because of its openness, and its proximity to a residential zone which it also factors in. K-means algorithms have a complexity of O(NTK), which is perfect because of its simplicity and the fact that it can efficiently scale well with large data sets which is important when theoretically the algorithm will have to deal with hundreds of islands, thousands of users, and multiple towers.

Zhao, Y. and Zhou, X. (2021)

IOPscience, *Journal of Physics: Conference Series*. Available at: https://iopscience.iop.org/article/10.1088/1742-6596/1873/1/012074 (Accessed: 14 December 2023).

Machine learning

Applying techniques we've learned in class, we could integrate Machine Learning algorithms that will use predictive models to forecast signal strength variations due to environmental factors like weather. natural disasters, etc. For example, if there's heavy rainfall that is predicted to affect Islands 'A' and 'B', the model will suggest boosting the signal strength in advance so internet connection isn't affected.

Pseudocode is on the next page ->>

Pseudocode

main.c

```
Algorithm OptimizeCommunciatonNetworks

Inputs: islandNetwork, topographyData, existingTowers, userDenistyData
Outputs: optimzedTowerPlacement, signalStrengthMap, networkFlowMap

Main()
    towerLocations = DetermineTowerLocations(islandNetwork, topographyData, existingTowers)
    networkFlowMap = CalculateNetworkFlow(islandNetwork, towerLocations)
    signalStrengthMap = GenerateSignalStrengthMap(towerLocations, userDensityData)
    DisplayResults(towerLocations, networkFlowMap, signalStrengthMap)

DetermineTowerLocations(islandNetwork, topographyData, existingTowers)
    towerLocations = empty list
    For each island in islandNetwork
        bestLocation = FindBestLocationForTower(island, topographyData, existingTowers)
        Add bestLocation to towerLocations
Return towerLocations

FindBestLocationForTower(island, topographyData, existingTowers)
    potentialLocations = EvaluateLocations(island, topographyData)
    highDensityAreas = ClusterUserDensity(island, userDensityData)
    bestLocation = ChooseOptimalLocation(potentialLocations, highDensityAreas)
Return bestLocation

CalculateNetworkFlow(islandNetwork, towerLocations)
    communicationGraph = CreateGraph(towerLocations)
    networkFlow = ApplyNetworkFlowOptimization(communicationGraph)
Return networkFlow

GenerateSignalStrengthMap(towerLocations, userDensityData)
    signalStrengthMap = CreateEmptyMap()
    For each tower in towerLocations
        coverageArea = SimulateSignalPropagation(tower, userDensityData)
        UpdateSignalStrengthMap(signalStrengthMap, coverageArea)
Return signalStrengthMap

DisplayResults(towerLocations, networkFlowMap, signalStrengthMap)
    ShowTowerLocations(towerLocations)
    ShowNetworkFlow(networkFlowMap)
    ShowSignalStrength(signalStrengthMap)
```

Conclusion:

This algorithm is capable of addressing the challenge of establishing reliable internet connection in a unique geographical location such as the Sea of Islands. By combining things like network flow optimization, graph theory, data driven placement strategies, and predictive machine learning models ensures that strong and easily adaptable connectivity is correctly tailored to the geographical and topographical diversity of these islands

3. (Marc Hayashi) Describe an algorithm that implements a Reinforcement Learning algorithm that optimizes the trading route based on emissions, travel time, and cargo distribution efficiency.

My goal is to utilize a machine learning algorithm to optimize trading routes based on emissions. To optimize the routes, I will implement a Reinforcement Learning algorithm called Proximal Policy Optimization (PPO).

-PPO is a reinforcement learning algorithm that is known for its effectiveness and simplicity. PPO addresses some challenges in policy-based RL models which makes it the best choice of algorithm. In reinforcement learning, policy optimization algorithms seek to learn a policy that maximizes cumulative rewards in an environment. The policy is represented by a neural network that is trained to learn actions to increase its reward. One of the major challenges in training is the balancing of the learning rate. If the adjustments are too drastic, the previous behaviors that were beneficial could be forgotten. However, if the learning is too slow, the model will be inefficient. PPO attacks this problem by controlling the amount that the policy is allowed to change at each update step. It uses a concept called "trust region" to ensure that the new policy is not too far from the old policy.

PPO improves on other RL methods like Q-learning, "vanilla" policy gradient methods, and trust region/natural policy gradient methods by implementing a more robust, efficient, and reliable algorithm. This method utilizes clipped probability ratios, which "performs a pessimistic estimate of the performance of the policy" and optimizes the policy by alternating between sampling data from the policy and performing several epochs of optimization on the sampled data (John Schulman).PPO also allows for continuous control tasks which will be perfect for my task to allow for continuous action inputs for the state of the trading vessels.

In policy gradient methods, an estimator of the policy gradient is computed and plugged into a stochastic gradient ascent algorithm which is an optimization algorithm that works to maximize a function.

$$\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \hat{A}_t \right]$$

Here, "π_θ is a stochastic policy and is an estimator of the advantage function at timestep t. E_t , the expectation, indicates the empirical average over a finite batch of samples, in an algorithm that alternates between sampling and optimization" (John Schulman). The estimator, g , comes from differentiating this function:

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t \left[\log \pi_\theta(a_t \mid s_t) \hat{A}_t \right].$$

The key here is to limit the frequency of the optimization step to prevent "destructively large policy updates".

PPO also utilizes a clipped objective function where the surrogate objective is modified to prevent the policy from destructive updates.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t\left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right]$$

Here, are the parameters of the policy, $r_t(theta)$ is the probability ratio of the action from the new policy (π_θ) to the old policy ($\pi_{\theta old}$). Epsilon is a hyperparameter that adjusts the amount of clipping. This clipping function works by taking the minimum of the clipped and unclipped objectives. If the policy change is within range $[1 - epislon,\ 1 + epsilon]$, the unclipped objective is used whereas if it is outside, the clipped objective is used. This limits how much the policy can update in one step.

The PPO algorithm uses the clipped objective function instead of LPG and performs multiple steps of SGA on it. In our case, we will be combining a loss function and a value function error term with the clipped objective function to work with a neural network so that the parameters can be shared between the policy and value function. The algorithm from the OpenAI PPO paper is shown below:

Algorithm 1 PPO, Actor-Critic Style

for iteration=$1, 2, \ldots$ **do**
  for actor=$1, 2, \ldots, N$ **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
    Compute advantage estimates $\hat{A}_1, \ldots, \hat{A}_T$
  end for
  Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leq NT$
  $\theta_{\text{old}} \leftarrow \theta$
end for

I will also need to create an environment that simulates the trade routes and ports in the pacific.

-To create a custom environment, we need to initialize the environment's state and define the action and observation spaces. The action will be defined on which island to go to, and the state will be the current node or island that the ship is at. This is limited by the directed graph as each node will only be able to travel to specific islands. The state will also include information on the amount of cargo, the total emissions of the trip, and the current time of the trip. The emissions will be based on the Greenhouse Gas calculation with each edge having a set value. This value will be added to the emission score of the trip and stored. The environment will also include the step which in this case is the transition to the next island. When the step is taken, the state is updated. The reward is also defined in the environment which will be defined based on the emissions and time it takes to distribute the trade goods. The emissions reward will be a negative value to serve as a penalty. We can calculate a total

reward from these two by scaling each value and combining them. The reward value will be normalized and returned with the state to the agent which chooses the next action.

In conclusion, this solution is an efficient and effective way to find optimal trade routes that minimize emissions while maximizing the distribution of trade goods. By utilizing PPO and a custom environment, the model can learn and plan out the best possible route of trade. The environment and its simple yet effective reward function combined with the effectiveness of PPO will help boost learning reliability and ability.

References

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov. "Proximal Policy Optimization Algorithms." Research Paper. Aug, 28 2017. https://arxiv.org/pdf/1707.06347.pdf.

OpenAI. *Gymnasium Documentation*. 2023. <https://gymnasium.farama.org/>.

4. Juvy Ann Lucero: Describe an algorithm that focuses on sustainable development and preservation of natural resources. We will utilize algorithms to map biodiversity hotspots based on ecological data (endemic species or habitat diversity). As we assign higher weights or sensitivity values to certain areas. The algorithm should modify the route to avoid or minimize routes passing through sensitive ecological zones whether traveling on land or on water. Implementing restricted or controlled access to ecologically sensitive areas by suggesting alternate routes or limited visitation. It should be able to balance tourism and resource distribution while prioritizing the conservation of natural resources and sensitive ecosystems across the islands. Continuous monitoring and adaptive strategies are crucial for their effectiveness in promoting sustainability and preserving the natural environment.

Objective:
Representing the islands, we are collecting data including all biodiversity information, endemic species, habitat diversity, and sensitivity values for different areas. Involve local communities in the preservation efforts and educate tourists about the importance of protecting sensitive ecosystems. Each approach relies on graph theory, optimization algorithms and dynamic updating mechanisms tailored to specific problems concerning navigation, resource distribution, tourism, and sustainability across the islands. These algorithms can adapt to changing conditions

and evolving requirements for the benefit of the island communities. We can construct an algorithm also prioritizing routes that minimize environmental degradation that include low-impact activities or that are very far away from the restricted areas. Low-impact activities include hiking or bird watching that will align with tourists' interest.

Graph Representation:
The islands and sensitive ecological zones can be represented as nodes, and the connections or paths between them are known as edges in a weighted graph. The weight of each edge could signify the sensitivity value or ecological impact.

Choice of Algorithm:
Dijkstra's Algorithm: We can utilize Dijkstra's algorithm that's known for finding the shortest path from one island to another in a weighted graph which is perfect in this situation because of the weighted map of ecological sensitivity. We can assign higher weights to zones that are the most sensitive in order for the algorithm to suggest alternative routes or restricting visitation through controlled access points. This algorithm can track multiple locations of land or sea that need to be off-limits to the public in the Polynesian triangle while looking out for multiple factors such as heavy tourism. It should prioritize places that have urgency for restoration the most based on how big it is (for example coral reefs) or are in need of protected wildlife habitats and favoring paths with lower sensitivity values. Dijkstra's algorithm is able to find the most efficient shortest paths to avoid and calculate these routes. By using Dijkstra's algorithm it will calculate routes and minimize impact on sensitive ecological zones while also considering multiple factors like travel time. The use of Priority Queues can be able to select paths, favoring lower total weights considering travel and sensitivity. Below will be a pseudocode where the 'sensitivity_weight_factor' could be a multiplier that increases the impact of sensitivity weights on path calculation for highly sensitive zones. This factor can be determined based on the degree of sensitivity or urgency for preservation.

Pseudocode:
```
alt = dist[u] + weight(u, v)

// Adjusting for sensitivity
if sensitivity[v] is high:
    alt = alt * sensitivity_weight_factor
```

Clustering Algorithms
We can utilize machine learning algorithms for hotspot identification with clustering algorithms. Clustering algorithms can be able to identify clusters representing biodiversity hotspots based on habitat diversity. By pinpointing biodiversity hotspots, conservation efforts can be targeted towards these areas for protection and needed resources. Reinforcement learning is a suitable fit for mapping biodiversity hotspots and optimizing routes to minimize impact on sensitive zones.

RL has the ability to learn from interactions with an environment, adapt to changes, and optimize decisions to achieve long-term goals. The continuous monitoring and adaptive strategies required for sustainability align with RL's capability to learn and adjust its behavior over time.

Minimum Spanning Trees

Graph-based algorithms can be utilized best to identify connectivity and priority areas of conservation. We can use minimum spanning trees application to determine connections between biodiversity hotspots for species movement. MST algorithms can find the shortest path to connect all nodes in a graph without creating cycles. As we know that highest sensitivity values imply greater ecological importance, we can start to define edge weights and values. Redirecting the path away from highly sensitive areas by finding alternative connections or paths while also suggesting controlled access to particular nodes will connect them together.

Pseudocode:

```
function KruskalAlgorithm(graph, ecologicalData,
ecologicalThreshold):
    sortEdgesBySensitivity(graph, ecologicalData)
    initialize MST as an empty set
    initialize cumulativeSensitivity = 0

    for each edge in sortedEdges:
        if adding edge to MST does not form a cycle:
            add edge to MST
            update cumulativeSensitivity += ecologicalData[edge]

        if cumulativeSensitivity >= ecologicalThreshold:
            break

    return MST

function ModifyRoute(originalRoute, MST):
    modifiedRoute = originalRoute

    for edge in MST:
        replace part of modifiedRoute passing through edge with
alternative route

    return modifiedRoute
```

The MST algorithm will connect nodes within each island to form MSFs, minimizing sensitivity values within these islands. Then, integrating these MSFs will form the optimal route network across islands, balancing conservation and travel needs. Modify the routes by identifying the overlaps or connecting points between MSFs to create a comprehensive route that minimizes the impact on sensitive zones. This is Kruskal's algorithm where routes following the edges in the MST will avoid paths passing through high sensitive zones.

Running Time:

As this is a draft, these are the main types of algorithms that aim to minimize impact, identify hotspots, and facilitate conservation on these islands which will make a positive impact throughout the years on. Integrating these algorithms can form a great strategy for sustainable development and an evenly conservation in the Sea of Islands. The running time of this algorithm can vary significantly based on several factors. The number of islands and connections will impact computation time. Computing sensitivity values and dynamically updating them may require additional computational resources. In its basic form, Dijkstra's algorithm has a time complexity of $O(V^2)$ with an adjacency matrix or $O((V + E) \log V)$ with an adjacency list, where V is the number of vertices (locations) and E is the number of edges (connections). Kruskal's algorithm running time would be $O(E \log E)$ or $O(E \log V)$ with efficient data structures (where E is the number of edges and V is the number of vertices). The overall complexity will depend on the size of the graph (number of nodes and edges), the clustering algorithm's performance, and the structure of the data.