C950 Documentation

The Dynamic Algorithm

The algorithm I used for my project is commonly known as a dynamic approach algorithm. The basic idea for this algorithm is to break down a large task into many smaller tasks that are individually much faster and easier to complete. Effectively, to determine the best list for directions, my algorithm looks for the shortest distance between two of the addresses, and pairs them together. It continues to pair up the shortest distances until it can complete a route through all addresses.

Overview of the Program

Algorithm Pseudocode

```
Input: a list of addresses, along with distance matrices for all other addresses
List of addresses = inList
Distance matrix = dDict
pairWithShortest() function (Finds shortest other address, and pairs as a list)
While len(inList) > 1:
        For each in inList:
                newPair = pairWithShortest(each)
                inList.append(newPair)
                inList.remove(each, newPair[-1])
Return inList[0]
```

Programming Environment

The programming environment used for this project was Visual Studio Code, version 1.65, with a 64-bit python version 3.9.10. I personally prefer this software due to its flexibility and debugging features. It also hosts a large variety of useful plugins for most languages.

This was run on a machine running windows 10.

Complexity of the program

Individual Functions:

- dataReset - Due to the function only being used to clear a number of lists, the big O notation is O(n)
- insertPackage - Due to the nature of appending to a list, this function has a complexity of O(1)
- updatePackage - Searching and updating a singular item in a list has a complexity of O(n)
- returnPackage - Searching the list has a complexity of O(n)
- dynamicProgrammingApproach - Due to the looping nature of the program, the time complexity of the program is O(n^3), which while better than brute force O(n!), likely has plenty of room for improvement.
- addressIdFind - Search algorithm for a list - O(n)
- addressFind - Search algorithm for a list, but with a different search parameter - O(n)
- load_Addresses - due to multiple nested loops, the big O of this one is O(n^2)
- load_Packages - no nested loops, just a single loop for basic searching. Big O: O(n)
- milesToTime - basic equation. Big O: O(1)
- timeToMiles - basic equation. Big O: O(1)
- distributePackages - Due to a loop within a loop nature of this one, it has a complexity of O(n^2)
- findShortestTruck - A simple search algorithm O(n)
- givePackageToTrucks - a single loop, thus simply O(n)
- deliverPackagesOnRoute - By far the least efficient function. 3 loops nested in a loop. O(n^4)
- Main - Technically has a big O of O(n) due to a loop, but as this is the GUI, it is unlikely to actually take anything like that amount of time.

The whole program:

The entire program has 3 main segments:

1. A search function for individual packages. This function, as stated before, has a big O average of O(n), as it is simply searching down a list and returning what it finds.
2. A return all packages function to display all packages for the day. This function simply prints out the list. As such, it has a big O of O(1)
3. An update time function. This updates the entire program, and delivers packages based on the entered time. Due to the deliverPackagesOnRoute function, the big O of this functionality is O(n^4)

Scalability

This program scales reasonably well, though the delivery function could likely be made faster to better increase speed. That said, individual package searches would scale very well. Similarly, the algorithm used to find the fastest path to each destination is significantly faster than brute force and would scale rather well.

Maintainability

The program has plenty of documentation and commentary in the code itself, to allow for easy navigation and access to the various functions. Similarly, due to the modular nature of the code, debugging problems and upgrading in the future would be very simple.

The self adjusting data structure: Pros and Cons

Pros:

- The hash table used is easily searched, and is readily readable without the need for further machine interpretation.
- Due to the format of the hash table, it would be relatively easy to send reports or similar from data based in it
- There is a secondary data structure to store package information tied to address information. This is used to ensure that no data is lost during processing, and to help optimize the complexity of the program itself

Cons:

- Due to there being some redundant data structures, the space required to store such information is doubled.
- If something is incorrect with one of the data tables, while it is relatively simple to spot that an error has occurred, determining which set to be correct can be made difficult.

Original Program

Attached along with document

The Data Structure

There are four primary data structures used in this program.

1. Package Class List
   a. Used for initial formatting of the package information. Also used to connect packages directly to their address information and the distance matrix
2. Address Class List
   a. Connected to each package is an address class object. This helps organize and optimize the distance matrix used for computation of the algorithm
3. Van Class List
   a. This is used to hold indexes of various packages, along with a total distance driven.
4. Package Hash List
   a. This is used to format the data from the Package class list into a readable and easily searchable format.

Screenshots

Packages at 9:

Packages at 10:

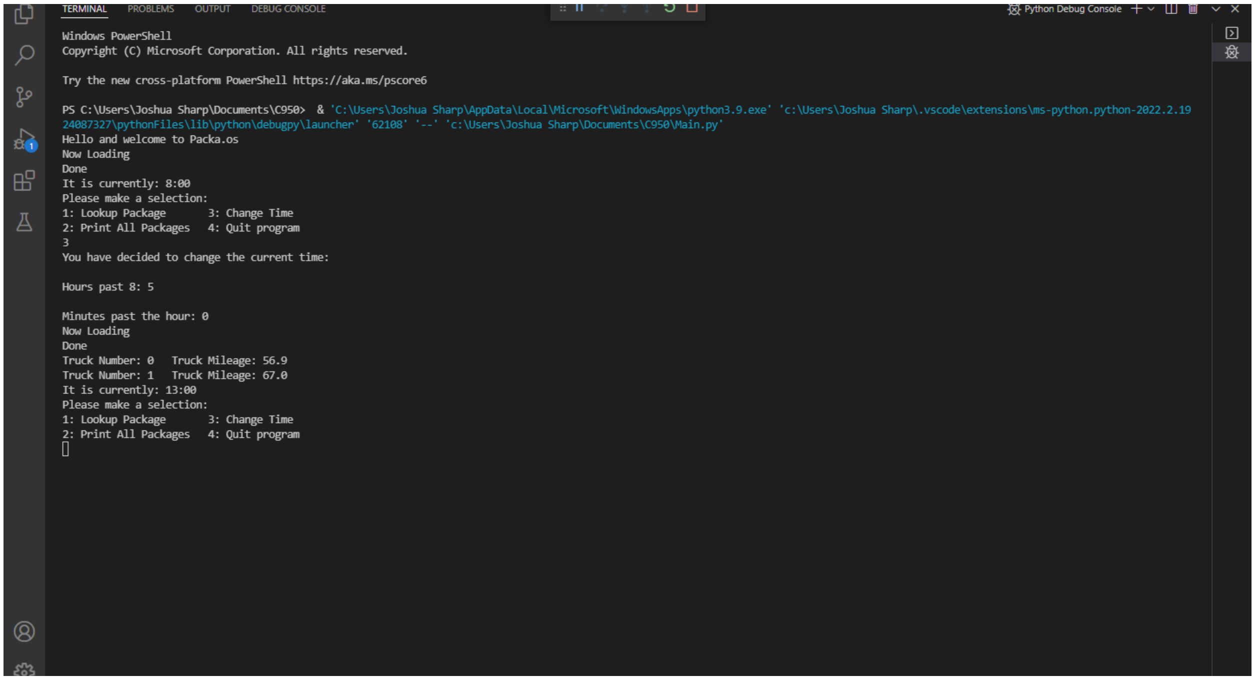
Joshua Pierce - WGU C950 Documentation

```
1: Lookup Package      3: Change Time
2: Print All Packages  4: Quit program
2
{'id': 1, 'address': '195 W Oakland Ave', 'deadline': '10:30:00', 'city': 'Salt Lake City', 'zip': '84115', 'weight': 21, 'status': 'delivered at: 8:15'}
{'id': 2, 'address': '2530 S 500 E', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84106', 'weight': 44, 'status': 'delivered at: 8:20'}
{'id': 3, 'address': '233 Canyon Rd', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84103', 'weight': 2, 'status': 'delivered at: 8:44'}
{'id': 4, 'address': '380 W 2880 S', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84115', 'weight': 4, 'status': 'delivered at: 8:14'}
{'id': 5, 'address': '410 S State St', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84111', 'weight': 5, 'status': 'delivered at: 8:41'}
{'id': 6, 'address': '3060 Lester St', 'deadline': '10:30:00', 'city': 'West Valley City', 'zip': '84119', 'weight': 88, 'status': 'at the hub'}
{'id': 7, 'address': '1330 2100 S', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84106', 'weight': 8, 'status': 'delivered at: 8:25'}
{'id': 8, 'address': '300 State St', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84103', 'weight': 9, 'status': 'delivered at: 8:44'}
{'id': 9, 'address': '300 State St', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84103', 'weight': 2, 'status': 'at the hub'}
{'id': 10, 'address': '600 E 900 South', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84105', 'weight': 1, 'status': 'delivered at: 8:35'}
{'id': 11, 'address': '2600 Taylorsville Blvd', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84118', 'weight': 1, 'status': 'delivered at: 9:36'}
{'id': 12, 'address': '3575 W Valley Central Station bus Loop', 'deadline': 'EOD', 'city': 'West Valley City', 'zip': '84119', 'weight': 1, 'status': 'en route'}
{'id': 13, 'address': '2010 W 500 S', 'deadline': '10:30:00', 'city': 'Salt Lake City', 'zip': '84104', 'weight': 2, 'status': 'delivered at: 8:58'}
{'id': 14, 'address': '4300 S 1300 E', 'deadline': '10:30:00', 'city': 'Millcreek', 'zip': '84117', 'weight': 88, 'status': 'delivered at: 9:57'}
{'id': 15, 'address': '4580 S 2300 E', 'deadline': '09:00:00', 'city': 'Holladay', 'zip': '84117', 'weight': 4, 'status': 'en route'}
{'id': 16, 'address': '4580 S 2300 E', 'deadline': '10:30:00', 'city': 'Holladay', 'zip': '84117', 'weight': 88, 'status': 'en route'}
{'id': 17, 'address': '3148 S 1100 W', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84119', 'weight': 2, 'status': 'delivered at: 9:19'}
{'id': 18, 'address': '1488 4800 S', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84123', 'weight': 6, 'status': 'delivered at: 9:38'}
{'id': 19, 'address': '177 W Price Ave', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84115', 'weight': 37, 'status': 'delivered at: 8:07'}
{'id': 20, 'address': '3595 Main St', 'deadline': '10:30:00', 'city': 'Salt Lake City', 'zip': '84115', 'weight': 37, 'status': 'delivered at: 8:06'}
{'id': 21, 'address': '3595 Main St', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84115', 'weight': 3, 'status': 'delivered at: 8:09'}
{'id': 22, 'address': '6351 South 900 East', 'deadline': 'EOD', 'city': 'Murray', 'zip': '84121', 'weight': 2, 'status': 'delivered at: 9:59'}
{'id': 23, 'address': '5100 South 2700 West', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84118', 'weight': 5, 'status': 'delivered at: 9:35'}
{'id': 24, 'address': '5025 State St', 'deadline': 'EOD', 'city': 'Murray', 'zip': '84107', 'weight': 7, 'status': 'at the hub'}
{'id': 25, 'address': '5383 South 900 East #104', 'deadline': '10:30:00', 'city': 'Salt Lake City', 'zip': '84117', 'weight': 7, 'status': 'at the hub'}
{'id': 26, 'address': '5383 South 900 East #104', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84117', 'weight': 25, 'status': 'en route'}
{'id': 27, 'address': '1060 Dalton Ave S', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84104', 'weight': 5, 'status': 'delivered at: 9:03'}
{'id': 28, 'address': '2835 Main St', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84115', 'weight': 7, 'status': 'at the hub'}
{'id': 29, 'address': '1330 2100 S', 'deadline': '10:30:00', 'city': 'Salt Lake City', 'zip': '84106', 'weight': 2, 'status': 'delivered at: 8:25'}
{'id': 30, 'address': '300 State St', 'deadline': '10:30:00', 'city': 'Salt Lake City', 'zip': '84103', 'weight': 1, 'status': 'delivered at: 8:42'}
{'id': 31, 'address': '3365 S 900 W', 'deadline': '10:30:00', 'city': 'Salt Lake City', 'zip': '84119', 'weight': 1, 'status': 'delivered at: 9:23'}
{'id': 32, 'address': '3365 S 900 W', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84119', 'weight': 1, 'status': 'at the hub'}
{'id': 33, 'address': '2530 S 500 E', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84106', 'weight': 1, 'status': 'delivered at: 8:20'}
{'id': 34, 'address': '4580 S 2300 E', 'deadline': '10:30:00', 'city': 'Holladay', 'zip': '84117', 'weight': 2, 'status': 'en route'}
{'id': 35, 'address': '1060 Dalton Ave S', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84104', 'weight': 88, 'status': 'delivered at: 9:03'}
{'id': 36, 'address': '2300 Parkway Blvd', 'deadline': 'EOD', 'city': 'West Valley City', 'zip': '84119', 'weight': 88, 'status': 'delivered at: 9:12'}
{'id': 37, 'address': '410 S State St', 'deadline': '10:30:00', 'city': 'Salt Lake City', 'zip': '84111', 'weight': 2, 'status': 'delivered at: 8:39'}
{'id': 38, 'address': '410 S State St', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84111', 'weight': 9, 'status': 'delivered at: 8:39'}
{'id': 39, 'address': '2010 W 500 S', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84104', 'weight': 9, 'status': 'delivered at: 8:58'}
{'id': 40, 'address': '380 W 2880 S', 'deadline': '10:30:00', 'city': 'Salt Lake City', 'zip': '84115', 'weight': 45, 'status': 'delivered at: 8:12'}
```

Packages at 1Pm:

```
1: Lookup Package      3: Change Time
2: Print All Packages  4: Quit program
2
{'id': 1, 'address': '195 W Oakland Ave', 'deadline': '10:30:00', 'city': 'Salt Lake City', 'zip': '84115', 'weight': 21, 'status': 'delivered at: 11:43'}
{'id': 2, 'address': '2530 S 500 E', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84106', 'weight': 44, 'status': 'delivered at: 8:20'}
{'id': 3, 'address': '233 Canyon Rd', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84103', 'weight': 2, 'status': 'delivered at: 8:44'}
{'id': 4, 'address': '380 W 2880 S', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84115', 'weight': 4, 'status': 'delivered at: 8:14'}
{'id': 5, 'address': '410 S State St', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84111', 'weight': 5, 'status': 'delivered at: 8:41'}
{'id': 6, 'address': '3060 Lester St', 'deadline': '10:30:00', 'city': 'West Valley City', 'zip': '84119', 'weight': 88, 'status': 'delivered at: 11:10'}
{'id': 7, 'address': '1330 2100 S', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84106', 'weight': 8, 'status': 'delivered at: 8:25'}
{'id': 8, 'address': '300 State St', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84103', 'weight': 9, 'status': 'delivered at: 8:44'}
{'id': 9, 'address': '300 State St', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84103', 'weight': 2, 'status': 'delivered at: 10:46'}
{'id': 10, 'address': '600 E 900 South', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84105', 'weight': 1, 'status': 'delivered at: 8:35'}
{'id': 11, 'address': '2600 Taylorsville Blvd', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84118', 'weight': 1, 'status': 'delivered at: 9:36'}
{'id': 12, 'address': '3575 W Valley Central Station bus Loop', 'deadline': 'EOD', 'city': 'West Valley City', 'zip': '84119', 'weight': 1, 'status': 'delivered at: 10:44'}
{'id': 13, 'address': '2010 W 500 S', 'deadline': '10:30:00', 'city': 'Salt Lake City', 'zip': '84104', 'weight': 2, 'status': 'delivered at: 8:58'}
{'id': 14, 'address': '4300 S 1300 E', 'deadline': '10:30:00', 'city': 'Millcreek', 'zip': '84117', 'weight': 88, 'status': 'delivered at: 9:57'}
{'id': 15, 'address': '4580 S 2300 E', 'deadline': '09:00:00', 'city': 'Holladay', 'zip': '84117', 'weight': 4, 'status': 'delivered at: 10:04'}
{'id': 16, 'address': '4580 S 2300 E', 'deadline': '10:30:00', 'city': 'Holladay', 'zip': '84117', 'weight': 88, 'status': 'delivered at: 10:04'}
{'id': 17, 'address': '3148 S 1100 W', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84119', 'weight': 2, 'status': 'delivered at: 9:19'}
{'id': 18, 'address': '1488 4800 S', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84123', 'weight': 6, 'status': 'delivered at: 9:38'}
{'id': 19, 'address': '177 W Price Ave', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84115', 'weight': 37, 'status': 'delivered at: 8:07'}
{'id': 20, 'address': '3595 Main St', 'deadline': '10:30:00', 'city': 'Salt Lake City', 'zip': '84115', 'weight': 37, 'status': 'delivered at: 8:06'}
{'id': 21, 'address': '3595 Main St', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84115', 'weight': 3, 'status': 'delivered at: 8:09'}
{'id': 22, 'address': '6351 South 900 East', 'deadline': 'EOD', 'city': 'Murray', 'zip': '84121', 'weight': 2, 'status': 'delivered at: 9:59'}
{'id': 23, 'address': '5100 South 2700 West', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84118', 'weight': 5, 'status': 'delivered at: 9:35'}
{'id': 24, 'address': '5025 State St', 'deadline': 'EOD', 'city': 'Murray', 'zip': '84107', 'weight': 7, 'status': 'delivered at: 10:19'}
{'id': 25, 'address': '5383 South 900 East #104', 'deadline': '10:30:00', 'city': 'Salt Lake City', 'zip': '84117', 'weight': 7, 'status': 'delivered at: 11:35'}
{'id': 26, 'address': '5383 South 900 East #104', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84117', 'weight': 25, 'status': 'delivered at: 10:03'}
{'id': 27, 'address': '1060 Dalton Ave S', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84104', 'weight': 5, 'status': 'delivered at: 9:03'}
{'id': 28, 'address': '2835 Main St', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84115', 'weight': 7, 'status': 'delivered at: 10:30'}
{'id': 29, 'address': '1330 2100 S', 'deadline': '10:30:00', 'city': 'Salt Lake City', 'zip': '84106', 'weight': 2, 'status': 'delivered at: 8:25'}
{'id': 30, 'address': '300 State St', 'deadline': '10:30:00', 'city': 'Salt Lake City', 'zip': '84103', 'weight': 1, 'status': 'delivered at: 8:42'}
{'id': 31, 'address': '3365 S 900 W', 'deadline': '10:30:00', 'city': 'Salt Lake City', 'zip': '84119', 'weight': 1, 'status': 'delivered at: 9:23'}
{'id': 32, 'address': '3365 S 900 W', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84119', 'weight': 1, 'status': 'delivered at: 11:15'}
{'id': 33, 'address': '2530 S 500 E', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84106', 'weight': 1, 'status': 'delivered at: 8:20'}
{'id': 34, 'address': '4580 S 2300 E', 'deadline': '10:30:00', 'city': 'Holladay', 'zip': '84117', 'weight': 2, 'status': 'delivered at: 10:04'}
{'id': 35, 'address': '1060 Dalton Ave S', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84104', 'weight': 88, 'status': 'delivered at: 9:03'}
{'id': 36, 'address': '2300 Parkway Blvd', 'deadline': 'EOD', 'city': 'West Valley City', 'zip': '84119', 'weight': 88, 'status': 'delivered at: 9:12'}
{'id': 37, 'address': '410 S State St', 'deadline': '10:30:00', 'city': 'Salt Lake City', 'zip': '84111', 'weight': 2, 'status': 'delivered at: 8:39'}
{'id': 38, 'address': '410 S State St', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84111', 'weight': 9, 'status': 'delivered at: 8:39'}
{'id': 39, 'address': '2010 W 500 S', 'deadline': 'EOD', 'city': 'Salt Lake City', 'zip': '84104', 'weight': 9, 'status': 'delivered at: 8:58'}
{'id': 40, 'address': '380 W 2880 S', 'deadline': '10:30:00', 'city': 'Salt Lake City', 'zip': '84115', 'weight': 45, 'status': 'delivered at: 8:12'}
```

Mileage of the trucks after all packages delivered:



Justification of the Core Algorithm

The algorithm I used is a Dynamic algorithm. The primary strengths of this algorithm are its complexity, which is to say that it is relatively simple to understand, and its efficiency, or how close it can get to the best possible solution.

The algorithm easily delivers all packages, with restrictions, before 140 miles is reached.

Two other algorithms I looked at for possible use are the Nearest Neighbor algorithm, which simply looks for the next nearest point until all the points are connected, and the brute force approach.

The nearest neighbor algorithm, while incredibly simple and fast, would likely not have been efficient enough to satisfy all restrictions. As such, I decided against its implementation.

Alternatively, an even simpler algorithm would have been to use brute force to check all possible combinations. While this would have been the best way to find the absolute most efficient route, I'm not positive my computer would have been able to complete the necessary computations before a few months had passed. As such, I decided against using it.

What I would do differently

The main thing I would do differently, is to start smaller. I would bother less with all possible restrictions, and just figure the project out, one step at a time, slowly building on the previous structures with a modular approach.

Justification of Data structures

Verification of requirements

The time needed to complete the lookup function is O(n), due to it being a simple list lookup. As such, time needed to find the appropriate information scales linearly with the total packages in the system. As such, this scales rather well as a data structure.

Due to the redundant nature of the data structure, for every 1 package, 2 packages worth of information is stored. This is not the most efficient use of space, but I believe it to be a worthy trade for the benefits to computation time this gives.

Due to each truck only stores the index of lists, and not lists themselves, the addition of more trucks or addresses would only affect time required for lookup linearly.

Other Data structures that could have been used

A case could be made for Tuples, as they are more space efficient than lists, and due to the nature of my program, tuples could have been used to safely store the information, so long as all package information was formatted and inserted at runtime.
Alternatively, a simply typed array could have been used, which would have provided the most efficient use of space. The tradeoff would be that all information would need to be the same type of data, such as Strings or similar. This would have made the handling of that data quite a bit more tricky on the computational side of things.