Access Where You Need It
University District Public Transit and Bicycle Paths

Samuel L. Peoples, CST, MA-C
July 18, 2016

Introduction

In this section, we will discuss the clients, contract, and projected cost of the project.

Seattle Metro and the University of Washington have petitioned the Mayor to increase access to public transportation to improve the quality of life of students and faculty living in the University District and the North Seattle area. Access to bus stops from bike routes was identified as a major concern from a census conducted in the fiscal year 2015 based on the ever-increasing volume of vehicles on campus. This issue was also evaluated and supported by the Environmental Science department, which noted the benefit of public transportation on Seattle's carbon footprint. The following table of high-volume bus stops (Seattle Metro) and associated image (Google Maps) at the end of this document (Fig. 1) display the desired bus stops to be included in the construction of the proposed bicycle path(s).

	Location
1	1520 24th Ave East
2	3712 Montlake Blvd NE
3	2315 NE 65th St
4	7000 35th Ave NE
5	2001 NE 75th St
6	6313 Latona Ave NE
7	3972 Eastlake Ave NE
8	2389 Boyer Ave E
9	2400 N 40th St
10	2900 Westlake Ave N
11	2807 3rd Ave W
12	52 West McGraw St
13	100 Mercer St
14	951 Mercer St
15	1441 E Aloha St

For the construction of this bicycle path, the cost of labor, closing certain streets, equipment maintenance, and assumed overtime will average the at $15,000.00 per mile of road covered. This value has been rounded up to the nearest thousand to ensure that the allotted budget is more than enough. The Mayor has instructed that the new bicycle paths will be painted over existing roads, due to details that are not relevant to the contract.

Modeling the Contract

Understanding of how the conclusions are made is essential. The models described in this section will support the associated findings and provide relevance to the contract.

The *Traveling Salesman Problem* was chosen to optimize this particular problem. This algorithmic approach can be summarized as a simulation where there exists a traveling salesman who

wants to start at a certain location, visit a set number of other locations, and then return to their starting location. Since each location can be accessed from every other location, the route he or she chooses to take, being the shortest route, might not immediately be available. This process of visiting each location once and returning to the starting location is called a *Hamiltonian Cycle*. In this instance, there are fifteen different bus stops, beginning with stop 1 and ending with stop 15. That means there are 15! or $1.31 \cdot 10^{12}$ different paths that could be painted.

Testing such a large number of possibilities can be quite difficult, and would require a lot of time. The number of possible paths can be reduced by creating something that is called a 'greedy' path, where the nearest location is used as the next location in the path. By following the immediately noticeable shortest path, the number of shorter paths than that is reduced to a much more manageable problem, which saves time and labor costs for when things are changed in the future. This is done by calculating the total distance to the next location, and removing it from the possibilities if the distance exceeds the 'greedy' value.

By applying the Traveling Salesman Problem to this contract, the optimal cost can be achieved, allowing for the least amount of construction to be done.

Solutions

Application of the previously described models can be found in this section. All distance approximations (and thus, associated costs) have been estimated based on optimized routes from Google Maps.

The graph at the end of this document (Fig. 2) provides a simple visual to see the possible connections between the bus stops. A more simplified display is below. The first row and first column is the bus stop, while the other cells denote the distance in miles to the cooresponding bus stops. Notice that the cells filled with zero are where the bus stop meets itself.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	1.2	3.1	3.6	4.7	3.9	2	0.8	2.7	4.1	4.7	4.9	3.9	3.3	0.9
2	1.2	0	1.9	2.5	2.6	2.8	1.7	1.8	1.6	3	3.6	3.9	4.3	3.7	2.8
3	3.1	1.9	0	0.8	0.7	1.2	2.1	3.3	2.7	4.7	5	4.8	5.9	4.9	4
4	3.6	2.5	0.8	0	1	2	2.9	3.6	3.9	5.2	5.8	5.5	6.9	6.3	4.6
5	4.7	2.6	0.7	1	0	1.5	2.8	4.1	2.9	4.4	5.3	5.1	5.9	5.3	5
6	3.9	2.8	1.2	2	1.5	0	1.7	2.9	1.5	3.7	4	3.8	4.9	4.6	4
7	2	1.7	2.1	2.9	2.8	1.7	0	1.2	1.5	3.1	4	3.8	4.9	4.7	4
8	0.8	1.8	3.3	3.6	4.1	2.9	1.2	0	1.9	3.3	3.8	4.1	3.2	2.6	1.8
9	2.7	1.6	2.7	3.9	2.9	1.5	1.5	1.9	0	1.6	2.5	2.3	3.4	3.2	3
10	4.1	3	4.7	5.2	4.4	3.7	3.1	3.3	1.6	0	0.7	0.9	2.3	1.7	3.4
11	4.7	3.6	5	5.8	5.3	4	4	3.8	2.5	0.7	0	0.5	1.7	2.4	4.1
12	4.9	3.9	4.8	5.5	5.1	3.8	3.8	4.1	2.3	0.9	0.5	0	1.2	2.2	3.6
13	3.9	4.3	5.9	6.9	5.9	4.9	4.9	3.2	3.4	2.3	1.7	1.2	0	0.8	2.4
14	3.3	3.7	4.9	6.3	5.3	4.6	4.7	2.6	3.2	1.7	2.4	2.2	0.8	0	1.7
15	0.9	2.8	4	4.6	5	4	4	1.8	3	3.4	4.1	3.6	2.4	1.7	0

The above table is used to find the 'greedy' path mentioned in the previous section.

Notice that in the row relating to bus stop 1, the shortest path is to bus stop 15, with a distance of .9 miles, from 15, excluding 1, the shortest path is to bus stop 14 with a distance of 1.7 miles. The following table follows that trend, providing a distance of 24.2 miles, equaling a necessary budget of $363,000.00. The graphic at the end of this document (Fig. 3) displays the path on a map.

1	15	14	13	12	11	10	9	6	3	2	7	8	4	5	1
0	.9	2.6	3.4	4.6	5.1	5.8	7.4	8.9	10.1	12.0	13.7	14.9	18.5	19.5	24.2

Note that the first row denotes the path taken, while the second row adds the total distance as that path is followed.

The path displayed above would have been easily found by someone without a mathematics background, and the method shown below clearly shows the most optimal path. Using the program Mathematica, a verified and accurate script was generated from previous work and applied to this problem.

```
SP = {{1}}; LP = {}; numNodes = 15; LPpathLengths = 0;
costListValue = {}; greedy = 242;
weights =
  {
   {0, 12, 31, 36, 47, 39, 20, 08, 27, 41, 47, 49, 39, 33, 09},
   {12, 0, 19, 25, 26, 28, 17, 18, 16, 30, 36, 39, 43, 37, 28},
   {31, 19, 0, 08, 07, 12, 21, 33, 27, 47, 50, 48, 59, 49, 40},
   {36, 25, 08, 0, 10, 20, 29, 36, 39, 52, 58, 55, 69, 63, 46},
   {47, 26, 07, 10, 0, 15, 28, 41, 29, 44, 53, 51, 59, 53, 50},
   {39, 28, 12, 20, 15, 0, 17, 29, 15, 37, 40, 38, 49, 46, 40},
   {20, 17, 21, 29, 28, 17, 0, 12, 15, 31, 40, 38, 49, 47, 40},
   {08, 18, 33, 36, 41, 29, 12, 0, 19, 33, 38, 41, 32, 26, 18},
   {27, 16, 27, 39, 29, 15, 15, 19, 0, 16, 25, 23, 34, 32, 30},
   {41, 30, 47, 52, 44, 37, 31, 33, 16, 0, 07, 09, 23, 17, 34},
   {47, 36, 50, 58, 53, 40, 40, 38, 25, 07, 0, 05, 17, 24, 41},
   {49, 39, 48, 55, 51, 38, 38, 41, 23, 09, 05, 0, 12, 22, 36},
   {39, 43, 59, 69, 59, 49, 49, 32, 34, 23, 17, 12, 0, 08, 24},
   {33, 37, 49, 63, 53, 46, 47, 26, 32, 17, 24, 22, 08, 0, 17},
   {09, 28, 40, 46, 50, 40, 40, 18, 30, 34, 41, 36, 24, 17, 0}
  };
```

The excerpt above contains the initial program input for this contract.

In Mathematica, the variables above were assigned to establish a baseline for creating the massive number of possible combinations for this contract. SP is the final list of paths, while LP is a working list, which is explained further below. The $numNodes$ is set to 15 to denote the number of bus stops, $LPpathLengths$ is initially set to zero, and $greedy$ is set to 242, the desired distance to beat. While the information provided by Seattle Metro and Google Maps was useful, to avoid programming errors, the values were mutliplied by ten to remove the decimals; this avoids some programming errors that may occur in the process of finding the optimal path.

```
(*BEGIN GIVEN CODE*)
While[LPpathLengths < numNodes, (
   For[i = 1, i ≤ Length[SP], i++, (
     For[j = 2, j ≤ numNodes, j++, (
        doAppend = True;
        For[k = 1, k ≤ Length[SP[[i]]], k++, (
          If[j == SP[[i, k]], (
             doAppend = False;
             Break[];)];)];
        If[doAppend == True, (
          AppendTo[LP, Append[SP[[i]], j]];)];)];)];
   LPpathLengths = Length[ LP[[1]] ];
   (*MODIFICATION TO GIVEN CODE*)
   For[i = 1, i < Length[LP], i++ (
      If[getCost[LP[[i]]] ≥ greedy, (
         LP = Delete[LP, i];)];)];
   (*END OF MODIFIED GIVEN CODE*)
   SP = LP; LP = {};)];
(*END OF GIVEN CODE*)
```

Above is the path builder provided from the Traveling Salesman template.

The path builder iterates through the existing paths, determines if the next value has been added to the list, and if it has not, it will add it to the list. Immediately following, it will check the cost of traveling that path. If it is greater than 24.2 miles, the entire path will be removed from the possible paths. This is essential to the optimization of the program because it will remove paths before they ever reach fifteen locations. Notice that the working set (LP) is used to determine the path that is being currently worked, while the final set (SP) contains each of the worked paths.

```
greedyIterator[alist_] := (
   For[i = 0, i < Length[alist], i++ (
       AppendTo[costListValue, getCost[alist[[i]]]])]
    Return[costListValue]);
```

This equation iterates through the working path and passes the data to the next equation below.

Using a *for loop*, populates a working list of different path costs which are shorter than 24.2 miles.

```
getCost[alist_] := (
   Sum[weights[[(alist[[i]]), (alist[[i + 1]])]], {i, 1, Length[alist] - 1}] +
    weights[[(Last[alist]), (alist[[1]])]]);
```

Referenced in the removal process and the *greedyIterator* equation, this short equation determines the cost of the path.

Using the embedded *Sum* function, the equation references the weights from the initial set of code based on the points provided from the passed in path. It then adds the distance from the final location to the cost to create a closed loop, ensuring that it is a *Hamiltonian Cycle*.

```
greedyIterator[SP];
Print["The shortest path is of length ", Min[costListValue], "."]
indexBest = Position[costListValue, Min[costListValue]];
Print["The paths are: "]
For[n = 1, n ≤ Length[indexBest], n++, (Print[SP[[indexBest[[n]]]]])];
```

The above code determines the solution and prints the data to the user.

The final section of the program passes the list of possible paths to the *greedyIterator*, populates the list of distances, and prints the output information. From the list of distances, the variable *indexBest* is created and set to the position of the lowest value, which helps the user find exactly which path is the optimal path.

Based on the output of the program, the following path was found; note that the first row denotes the path taken, while the second row adds the total distance as that path is followed.. The graphic at the end of this document (Fig. 4) displays the path on a map. After receiving the path from the program, the locations were put into Google Maps and an optimal path was found. Note that Google Maps has reduced the distances between some points, which would result in repainting paths that have already been completed.

1	15	8	7	2	9	6	3	4	5	10	11	12	13	14	1
0	.9	1.8	3.0	4.7	6.3	7.8	9.0	9.8	10.8	15.2	15.9	16.4	17.6	18.4	20.1

Observe that the cost associated with the optimal path is $301,500.00, a reduction of $61,500.00 from the initial greedy path.

Recommendation

The following recommendation is supported by the findings of the applied models and their benefit to the community. The client has no legal obligation to adhere to these findings.

It is in the best interest of the University of Washington, Seattle Metro, and the City of Seattle to base their construction around the optimized path. The savings of over $60,000.00 allows for additions to be made, connecting other bus stops to this new bicycle path. This bicycle path allows for students and faculty members to better access the campus, reducing the number of vehicles parked on campus, campus vehicle emissions, and improving the community through the promotion of exercise and fitness.

Based on these findings, an additional $28,000 could be allotted towards connecting the bus stops located at 3712 Montlake Blvd NE and 2315 NE 65th St, which still allows for $33,000 of excess budget left over from the greedy path.

Although the two generated paths are remarkably similar, it is evident in their cost that the more cost-effective decision would be the optimized path.

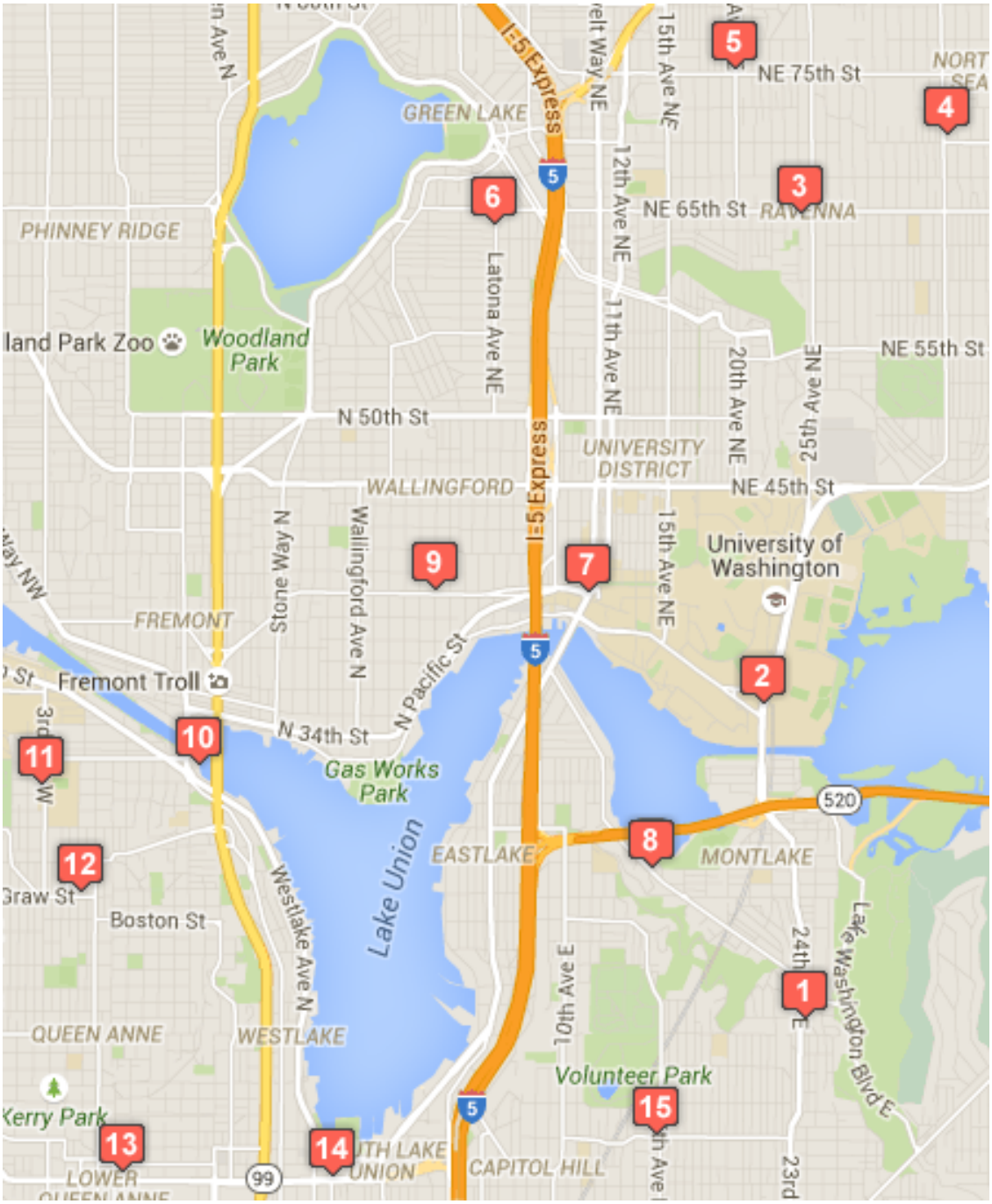

Figure 1: Map of bus routes provided by Seattle Metro, with assistance from Google Maps

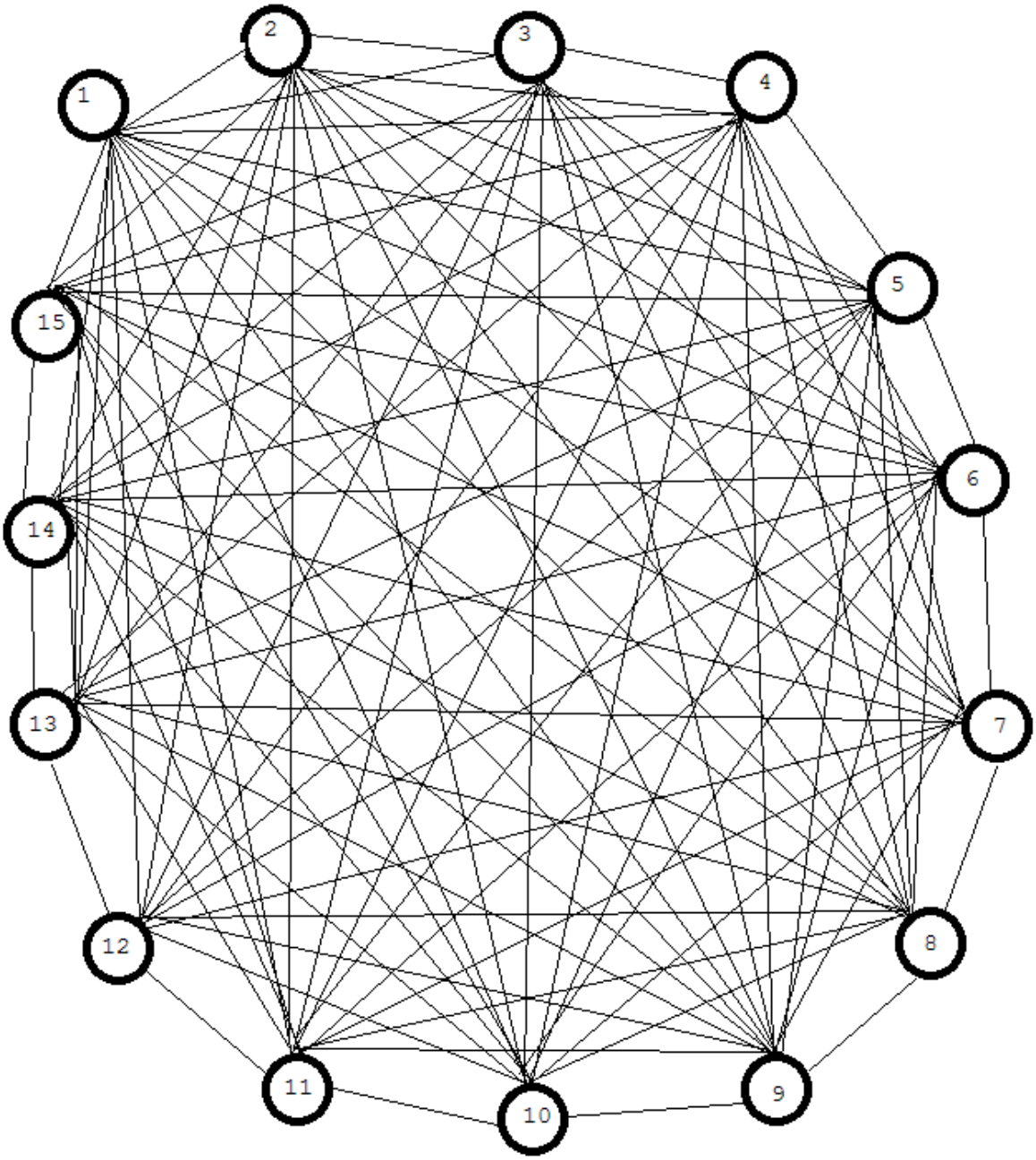

Figure 2: All possible connections of bus stops

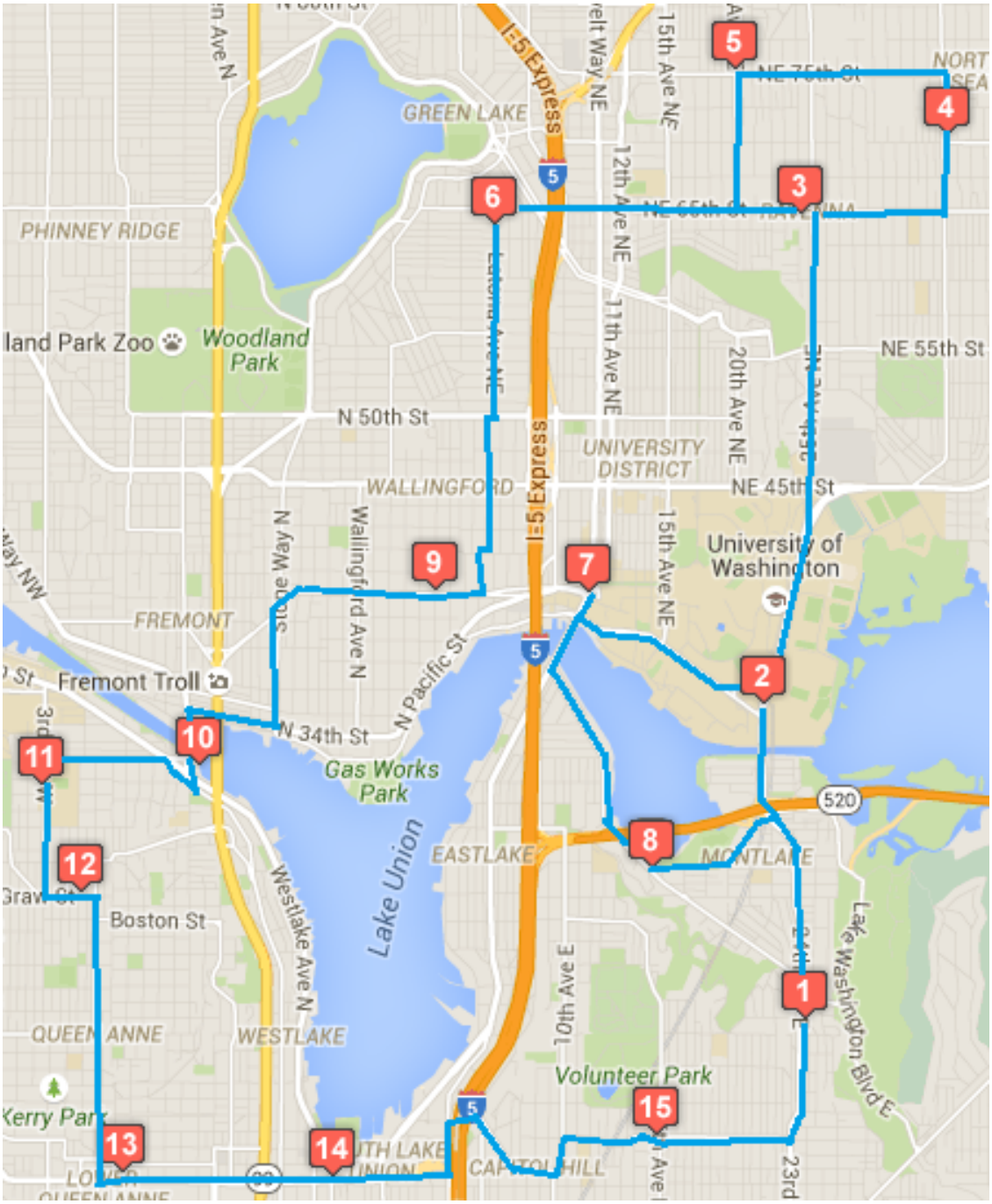

Figure 3: The greedy path chosen, with assistance from Google Maps

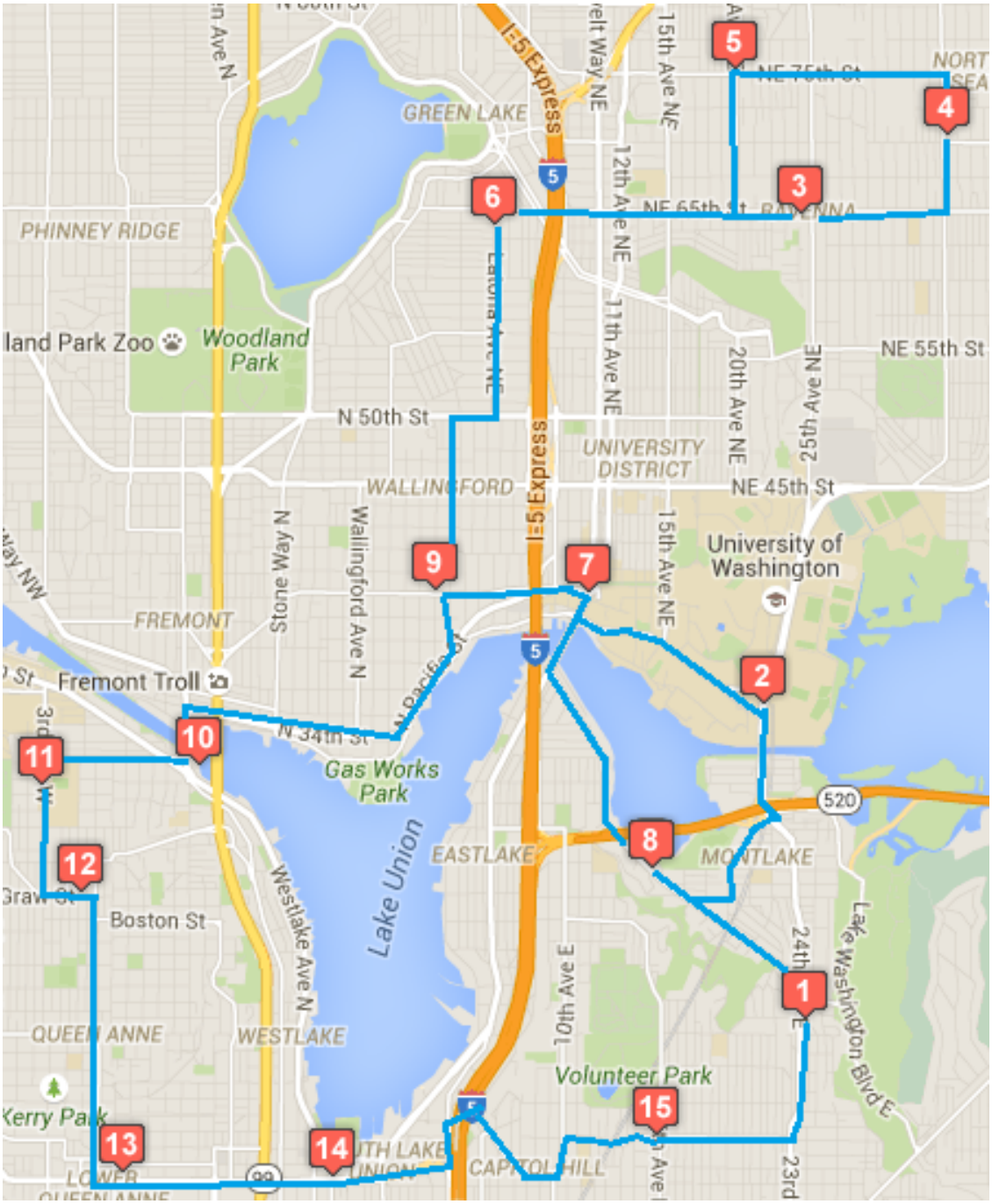

Figure 4: The optimized path, with assistance from Google Maps

This page intentionally left blank.