Paper 3875-2019

Using Proc Optgraph to implement the Prize Collecting Traveling Salesman Problem in SAS (Gotta catch as many as we can in a Pokémon raid for Alice)

Bryan Yockey and Joe DeMaio,
Department of Statistics and Analytical Sciences, Kennesaw State University;

ABSTRACT

The classic Traveling Salesman Problem (TSP) establishes a list of cities to visit and costs associated with travel to each location. The goal is to produce a cycle of minimum cost that visits each city and returns the salesperson to her home location. What happens if an imposed time limit on the journey makes visiting all locations impossible? Assuming each location is of equal value then the goal transforms into visiting as many locations as possible within the imposed time limit. This variation is known as the Prize Collecting Traveling Salesman Problem (PCTSP). We use Proc Optgraph to implement the TSP, develop an approach to implement the PCTSP and utilize SAS® VIYA® to map results. Set in a suburb of Metropolitan Atlanta, our motivation stems from the need to acquire as many virtual Pokémon as possible for Alice, the nine-year old daughter of one of the authors. Analysis of results from executing SAS® generated routes is included.

INTRODUCTION TO THE TSP

The traveling salesman problem (TSP) wishes to devise a route that starts a salesperson at a home base, visits every client on a list exactly once, and returns to the starting location. At times, the TSP is a question of existence such as a closed knight's tour of a chessboard, a classic problem in recreational mathematics. Can a knight use legal moves to visit every square on the board and return to its starting position? While originally studied for the standard 8×8 board, the problem generalizes easily to other rectangular boards. The 3×3 board does not admit a closed knight's tour as it is impossible to enter or exit the center square. The standard 8×8 board admits a closed knight's tour. Proc Optgraph trivializes the task of finding a closed knight's tour, should one exist.

```
*First, create the chessboard as a graph of legal knight moves;

data board;
rows = 8;
columns = 8;
do i = 1 to rows;
do j = 1 to columns;
origin =  (i || j);
/* input legal moves of the knight */
/* up 2 right 1 */
if i>=3 and j<= columns-1 then do
      destination = (i-2 || j+1);
      output;
      end;
/* up 1 right 2 */
if i>=2 and j<= columns-2 then do
      destination  = (i-1 || j+2);
      output;
      end;
/* down 1 right 2 */
if i<=rows-1 and j<= columns-2 then do
      destination  = (i+1 || j+2);
      output;
      end;
/* down 2 right 1 */
if i<=rows-2 and j<= columns-1 then do
      destination  = (i+2 || j+1);
      output;
      end;
end;
end;
run;

*Second, construct a TSP for the knight's graph;

proc optgraph
data_links = board;
*create graph from variables;
data_links_var
from = origin
to = destination;
* write cycle to file closed_knights_tour;
tsp out = closed_knights_tour;
run;
```

The closed knight's tour of the 8×8 board produced by this code is located in Appendix A. For a list of all rectangular chessboards that admit a closed knight's tour, see Schwenk's 1991 paper.

The closed knight's tour is a question of existence in a TSP. The knight cannot legally move from just any square to any other square. Furthermore, there is no cost associated with a knight's move. In other TSPs, widely varying costs are associated with possible travel between every pair of locations. Cost definitions include time, monetary expenses, or any other variable for optimization. In such problems, existence of a tour is not in question. Finding an optimal tour is the goal.

Robert Allison (2016) generates an optimal route for collecting rare Pokémon in North Carolina in his blog on SAS Learning Post as shown in Figure 1. Here the goal is to minimize total distance in miles of the tour. Reproducing the work to minimize travel time rather than distance, the route produced was only slightly different. The statewide scale and distance between locations produced likely similar results despite minimizing different variables.

Typically, graphs or networks model the TSP. In a transportation problem, the graph $G = (V,E)$ is an ordered pair of sets where the vertex set V represents a set of locations and the edge set E represents the minimum cost of traveling between any two locations. The TSP is known to be np-complete.

Obviously, it will take a few days to collect all of the rare Pokémon in North Carolina. What if a time limit exists for collecting Pokémon? One might not be able to collect every target on the list. You would then want a route that would capture as many targets or prizes as possible in the constrained time limit. In 1989, Egon Balas defined this as the prize collecting traveling salesman problem (PCTSP). The PCTSP is never a question of existence but always a question of maximizing the number of prizes collected in a fixed period. As with the TSP, the PCTSP is np-complete as well.

Like Robert Allison's (2016) blog example, our motivating problem collects Pokémon as well. A Pokémon raid is a very different undertaking than traveling to a location and collecting a Pokémon. Raids involve first defeating a Pokémon in battle; these battles, particularly the ones of interest for this project, require multiple players participating at the location, at the same time. If the players successfully defeat the raid boss, they acquire the opportunity to capture the Pokémon they defeated. This opportunity is the prize in our PCTSP.

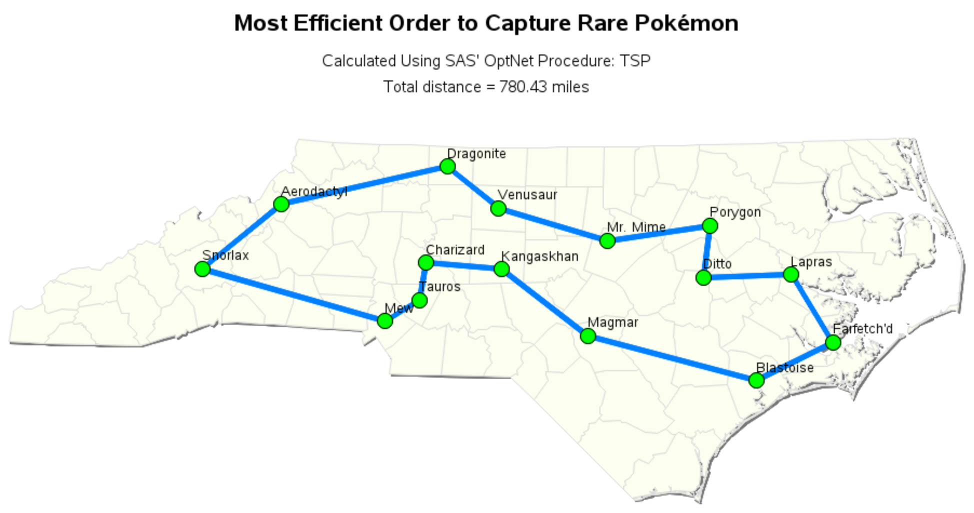
Most Efficient Order to Capture Rare Pokémon
Calculated Using SAS' OptNet Procedure: TSP
Total distance = 780.43 miles
Dragonite
Aerodactyl
Venusaur
Porygon
Mr. Mime
Snorlax
Charizard
Kangaskhan
Ditto
Lapras
Tauros
Mew
Magmar
Farfetch'd
Blastoise

Figure 1: TSP for capture of rare Pokémon in North Carolina

Every day, raid battles are generated randomly all over the world. They sporadically appear and last for 45 minutes. Interest in this project began in summer 2018 as Niantic, the game developer, rolled out a new variation of Pokémon Go. On specific days, every possible raid location has the same raid boss available for battle for three hours. Players can attempt the raid at each location until they defeat the boss once.

DETERMINING RAID EVENT DURATION

Needing to defeat a boss while working within a time constraint creates a unique situation when determining the number of locations that are possible to visit. As the process improves, the number of locations achievable increases which increases the total time to drive to battle the bosses. Thus, the amount of time remaining to travel between the locations decreases.

$$Total\ Travel\ Time + \Sigma\ Raid\ Event\ Durations\ \le 180\ minutes$$

When each raid event begins, a two-minute timer starts that allows other players to join. This portion of the event is fixed and cannot be shortened. When the timer ticks down to zero the battle begins. Players must defeat the boss Pokémon within 300 seconds. If the boss still has health after 300 seconds, the battle is a loss for the players. Skilled teams with upper level players will likely defeat a boss faster than the less experienced or smaller team. After defeating the boss, players get the opportunity to catch the prize Pokémon with a number of special Poké balls that is determined by a number of factors. An explanation of the number of balls for catching the prize Pokémon is in Appendix B.

Naturally, the first effort of time optimization began on the ground at a raid location. The catching phase lasts until the player catches the Pokémon or runs out of balls. The more balls needed, the longer the event lasts.   Catching the Pokémon is a function of skill and luck. A seasoned player has a better chance of ending the catching phase sooner by catching the Pokémon earlier. One member of the group was tasked with keeping the time of each event with a stopwatch. The timer began with the initiation of the raid and ended when the first player either caught the Pokémon or it ran away. Analysis found a duration time of 239 seconds. The distribution of times is highly skewed, so the median was a better measure of duration remaining for traveling between locations. More information on the calculation of raid duration time can be found in Appendix C. A player does not have to remain at the location while catching. Thus, a large portion of the variation in event time caused by the catching phase was mitigated by having a passenger in the car attempt to catch the driver's prize. As soon as the battle phase was over, the group would begin traveling to the next location. The team felt we had done everything to minimize the amount of time at each raid. Turning to software to maximize the number of locations seemed to be the next natural way to improve our results. Thus, we turned our attention to minimizing travel time to maximize the number of locations we visited. Knowing the location of events and a time limit necessitated a change in strategy. Figure 2 maps the 35 locations of raids for consideration within the three-hour time limit.

Niantic places gym locations at areas of interest, monuments, and public art displays. This project focuses on a team playing in Cartersville, Georgia. We omit locations for various reasons. These events take place on Saturdays; thus, we avoid gym locations at churches that meet on Saturday. Cemeteries are often full of gym locations. Many of the players in the group avoid the cemetery out of respect.   Parks also tend to be a treasure trove of gym locations. When festivals and local events are scheduled at parks, the group will exclude locations inside from the route as congestion of attendees slows the process. This is an easy adjustment to make, as finding information on events in a park in advance of the Pokémon Go event is easy. Our community of players tries to only play where we will not be disruptive. We maintain and respect a list of locations where our Pokémon activities might be disruptive.

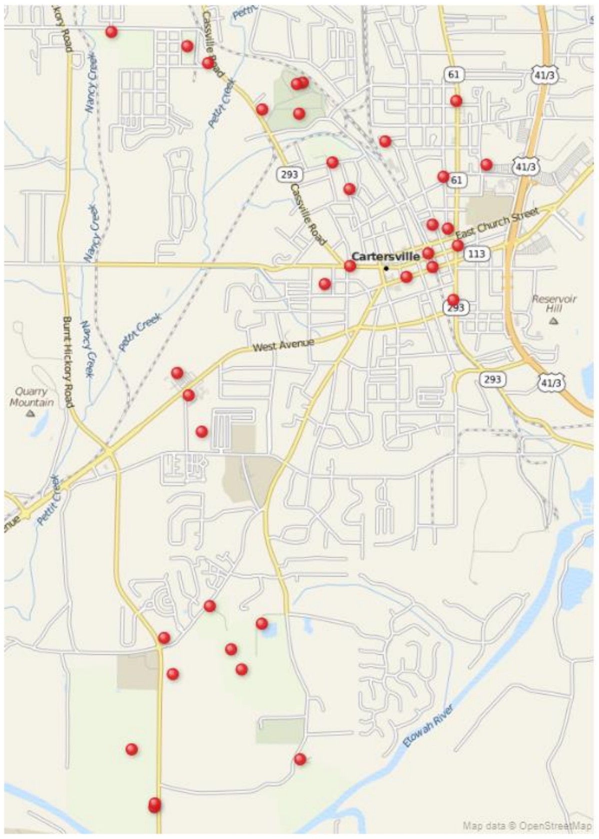

Figure 2: Map of all raid locations

ANALYSIS FREE AND GREEDY APPROACH

On the first raid, a team of seven people gathered the morning of the event and picked a starting location. We began our route at a park that contained three locations. With no prior planning or analysis and as the event unfolded, we decided by committee the best route. As one might imagine, making live decisions by committee may not lead to a strategically, consistent approach, let alone an optimized route. The group utilized a greedy approach by selecting the next closest location from its current position. When the three-hour timer expired, we had visited twenty-two locations. This seemed to be a successful day.

With some experience under our belt, a better plan formed for the second event. Slight adjustments came about by omitting a few locations due to circumstances such as needing to walk to a location from the parking lot, difficult parking situations, and avoiding traffic lights and train tracks. With these adjustments, we were able to increase our number of locations to twenty-five. This was better but still a greedy approach with no formal use of analytics.

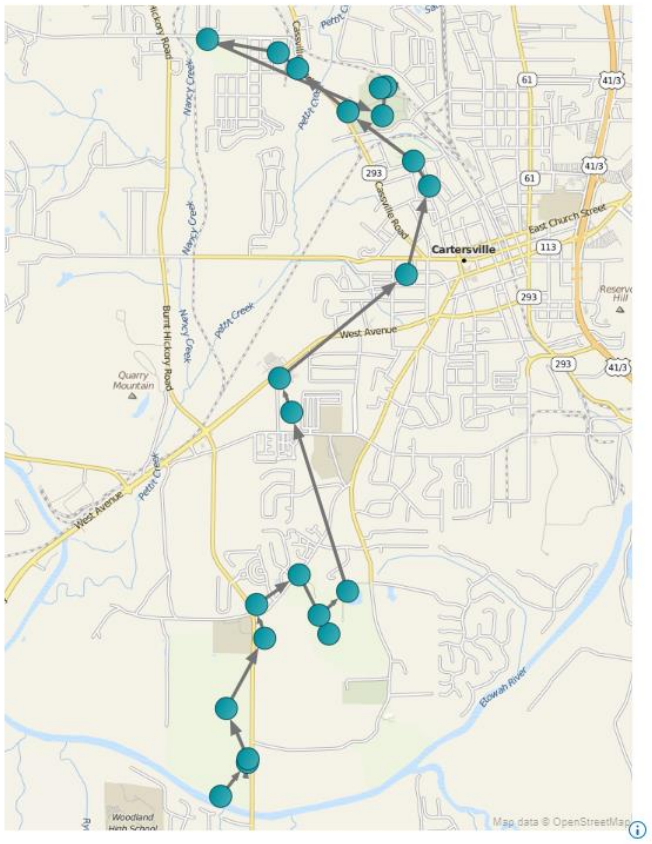

Figure 3: Map of Greedy Route

IMPLEMENTING THE TSP SOLUTION

Proc Optgraph does not specifically implement PCTSP. The first attempt to improve the route uses Proc Optgraph to design a traditional traveling salesman solution. The 37 local raid locations formed the vertex set of the complete graph. Each weighted edge in the graph represented the driving time between a pair of locations. Scraping Google Maps produced the driving time between locations. Proc Optgraph generated a tour of the 37 locations

```
*Graph Creation;
proc optgraph
data_links = times;
*create graph from variables;
data_links_var
from = home
to = next
weight=Time;
* write optimal cycle to file gym_time_TSPTour;
tsp out = gym_time_TSPTour;
run;
```

Working with a group of 37 locations, and thinking that 37 raids would be unattainable, a goal of thirty raids was set. This number was chosen since it was almost a 50% increase in locations from the initial attempt. To produce a 30-location route, we deleted the seven consecutive locations that had the longest drive time. We began traveling from the first location in the resulting path. Unfortunately, an eight-minute glitch in the Pokémon Go servers cut into the three-hour limit. Things progressed quickly after this delay. At a few points during the event, the SAS generated route took us to locations in an order that contradicted our intuition. The data scientist on the team had gathered a group of players who agreed to follow the route to the end of the three-hour limit though some grumbling ensued. The Proc Optgraph route allowed us to increase the number of prizes to 28. At the end, even the grumbling naysayers admitted the non-intuitive route was superior. Furthermore, at the end of the three-hour timer two more locations were within sight of the group. It seems likely that without the initial server issues the goal of 30 gym locations would have been achieved.

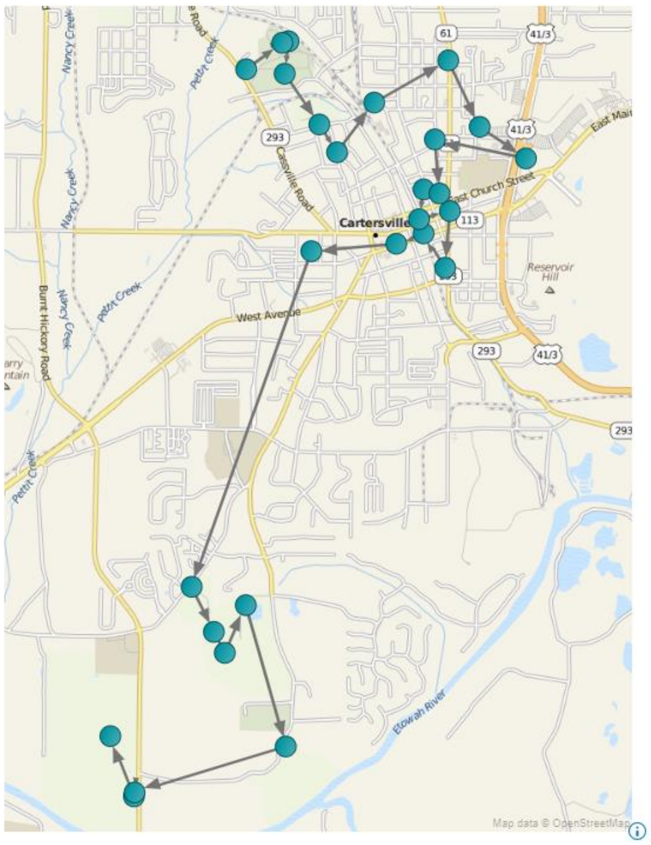

Figure 4: Map of TSP Route

IMPLEMENTING THE CLIQUE SOLUTION

Implementing a TSP route for the PCTSP problem seems like a good starting point. However, the TSP will likely make sacrifices to time when plotting a return route to the starting location. We wanted an approach that did not make such a sacrifice.

In graph theory, a clique is defined as a set of vertices, in this case gym locations, that are all adjacent and not part of a bigger such set. In an attempt to improve upon the modified TSP solution, cliques of locations were created using Proc Optgraph. The locations were grouped by combining locations into groups based on the maximum drive time between all the locations in the groups. The data scientist chose an upper bound of 120 seconds because this max drive time created a reasonable number of cliques, namely 12. These cliques were not disjoint. By hand, cliques were combined based on the number of overlapping locations. Nine disjoint cliques resulted from this process.

```
proc optgraph
 data_links = WORK.POKEMON_CLIQUES_graph;
 data_links_var
 from = home
 to = next;
 clique out = Cliques_in_Pokemon_graph;

*find all cliques in graph;
 title 'All Cliques';
run;
```

The locations of the cliques fall roughly in a linear form from southwest to northeast across Cartersville. It seemed natural to start at one end and continue through the cliques toward the other end. The weight to determine the order in which to visit the cliques was set by the minimum drive time between pairs of individual cliques. Starting at one end, the total drive time from the origin to all the cliques was calculated. The path with the shortest total drive time from one end to the other was then established. Once the optimal order to visit the cliques was determined, the cliques were stitched together into the final route. A TSP solution was implemented to create routes within the individual cliques. For every transition from one clique to the next, the data scientist had to decide at what point to enter the TSP loop and which direction to travel through it for the transition to the next. Having a good working knowledge of the area played a large role in the creation of this route. While using software to collect so many drive times, it becomes difficult to track the route that Google Maps is using to go between the locations. Shortcuts and lightly traveled roads were utilized that may have not been suggested otherwise.

Vertex	Shortest Time From A	Previous Vertex
A	0	
B	98	A
C	267	A
D	384	C
E	430	D
F	410	A
G	526	A
H	457	A
I	649	D

Table 1: Drive time to each clique from the beginning

When creating the cliques, two locations formed cliques of size 1. This implies that these two locations were far away from all other sites and removed from consideration. One of these locations was used in the modified TSP route, the other belonged to the section of the loop that was removed. From the original group of 37, there were 35 locations used for the implementation of the clique solution.

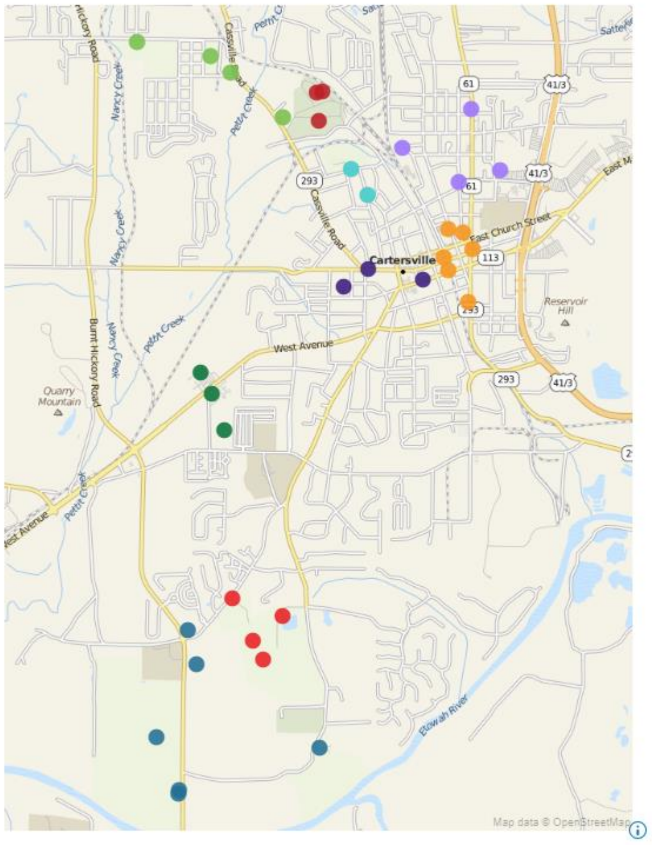

Figure 5: Map of Locations Colored by Clique

Unfortunately replicating the clique solution with the same group of players used for the TSP solution was not possible. While developing the clique solution, more gym locations had been added to the game. Players were unwilling to ignore new gyms in order to compare the different routes. Thus, a simulated run was used to assess the Clique Solution Route. For this simulation, the data scientist drove between locations, amidst the appropriate traffic conditions, and recorded actual drive time using a stopwatch. Utilizing these times, along with 239 seconds per location, determined the number of targets achievable. The loophole in this approach was that it did not take technical difficulty, like the server glitch in the TSP route or player congestion at a location, into account. This was accounted for by using the median raid event time. By implementing the clique solution, the team was able to visit 34 locations.

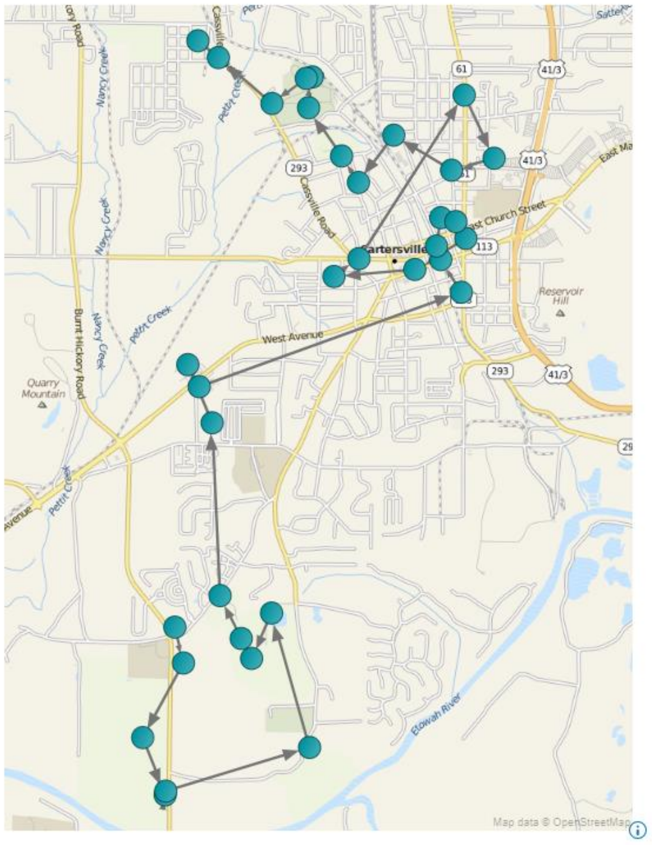

Figure 6: Map of Clique Solution Route

CONCLUSION

Conducting analysis in SAS® by Proc Optgraph increased the number of raids the team was able to complete by 55 percent. SAS® Viya® created detailed maps critical to understanding the geography of this problem specific to Cartersville, Georgia. The increase in the number of raids, from 22 to 34, from the greedy approach to the modified TSP and again with the Clique Approach illustrate the effectiveness of this process.

FUTURE WORK

SAS® Proc Optgraph provided important analysis of the traffic network for this problem. However, at certain junctures, the data scientist constructed portions of the results by hand. Determining the limiting time for clique construction and joining cliques to create a disjoint partition of sites are examples of such work. A clear next step for this work is automating such steps for wider and faster implementation.

REFERENCES

Allison, Robert (2016). Most efficient way to find rare Pokémon. Retrieved from https://blogs.sas.com/content/sastraining/2016/07/29/most-efficient-way-to-find-rare-pokemon/

Balas, E., (1989). The prize collecting traveling salesman problem. *Networks, 19*, 621–636.

Gross, J. L., Yellen, J., & Zhang, P. (2014). Handbook of graph theory: Edited by Jonathan L. Gross, Columbia University New York, USA; Jay Yellen, Rollins College Winter Park, Florida, USA; Ping Zhang, Western Michigan University, Kalamazoo, USA. Boca Raton: CRC Press.

Henshaw, H. M., Staples, L., & DeMaio, J. (2018). *Graph visualization for PROC OPTGRAPH*. Paper presented at South Eastern SAS Users Group 2018 conference, St. Petersburg, Florida. Retrieved from https://www.lexjansen.com/sesug/2018/SESUG2018_Paper-286_Final_PDF.pdf

SAS®, "SAS® OPTGRAPH Procedure." August 2018.

Schwenk, A. J. (1991). Which rectangular chessboards have a knight's tour? *Mathematics Magazine 64*(5), 325-332.

ACKNOWLEDGMENTS

I would like to thank the faculty in the Department of Statistics and Analytical Sciences for all the support and advice they provided me throughout this project. I would like to thank the Analytics and Data Science Institute for providing me the necessary software for this research. I would like to thank SAS Global Forum for hosting my work. I would like to thank the Kennesaw State University Graduate Student's Association for travel funding. I would also like to thank the group of friends I have made while playing this game. Last, but not least, I would like to thank my wife, Virginia, and Daughter Alice for their continued support of, and enthusiasm for this undertaking.

RECOMMENDED READING

- Watkins, J. J., & Rzewnicki, A. (2012). Across the board: The mathematics of chessboard problems. Princeton: Princeton University Press
- Catch Pokémon in the Real World with Pokémon GO! (n.d.). Retrieved March 08, 19, from https://www.pokemongo.com/en-us/

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Bryan Yockey
Kennesaw State University
byockey@students.kennesaw.edu

Joe DeMaio
Department of Statistics and Analytical Sciences
Kennesaw State University
jdemaio@kennesaw.edu

APPENDIX A: CLOSED KNIGHT'S TOUR ON THE STANDARD CHESSBOARD

Table 2 contains the knight moves generated in Proc Optgraph for the closed knight's tour on the 8 by 8 chessboard.

Move			**Move**			**Move**			**Move**		
1	1 1	2 3	**17**	4 7	2 8	**33**	1 4	2 6	**49**	8 7	6 8
2	2 3	1 5	**18**	6 6	4 7	**34**	2 2	1 4	**50**	7 5	8 7
3	3 4	1 5	**19**	5 4	6 6	**35**	4 1	2 2	**51**	8 3	7 5
4	5 3	3 4	**20**	4 2	5 4	**36**	4 1	3 3	**52**	7 1	8 3
5	5 3	4 5	**21**	6 1	4 2	**37**	2 1	3 3	**53**	7 1	5 2
6	6 4	4 5	**22**	6 1	8 2	**38**	2 1	1 3	**54**	3 1	5 2
7	7 2	6 4	**23**	8 2	7 4	**39**	1 3	2 5	**55**	3 1	1 2
8	7 2	8 4	**24**	6 2	7 4	**40**	2 5	1 7	**56**	1 2	2 4
9	8 4	6 5	**25**	8 1	6 2	**41**	1 7	3 8	**57**	4 3	2 4
10	6 5	4 6	**26**	8 1	7 3	**42**	5 7	3 8	**58**	4 3	5 5
11	4 6	2 7	**27**	7 3	8 5	**43**	5 7	7 8	**59**	5 5	3 6
12	2 7	4 8	**28**	8 5	7 7	**44**	8 6	7 8	**60**	4 4	3 6
13	5 6	4 8	**29**	7 7	5 8	**45**	8 6	6 7	**61**	6 3	4 4
14	3 5	5 6	**30**	3 7	5 8	**46**	6 7	8 8	**62**	5 1	6 3
15	3 5	1 6	**31**	3 7	1 8	**47**	7 6	8 8	**63**	5 1	3 2
16	1 6	2 8	**32**	2 6	1 8	**48**	7 6	6 8	**64**	1 1	3 2

Table 2: A closed knight's tour

Figure 7 appears in the 2018 paper by Henshaw, Staples, and DeMaio, Graph Visualization for PROC OPTGRAPH which describes a method for exporting SAS data to visualize a graph. Together the black and red lines indicate the edges in the knight's graph for the 8 by 8 board. The red lines show one possible closed knight's tour.

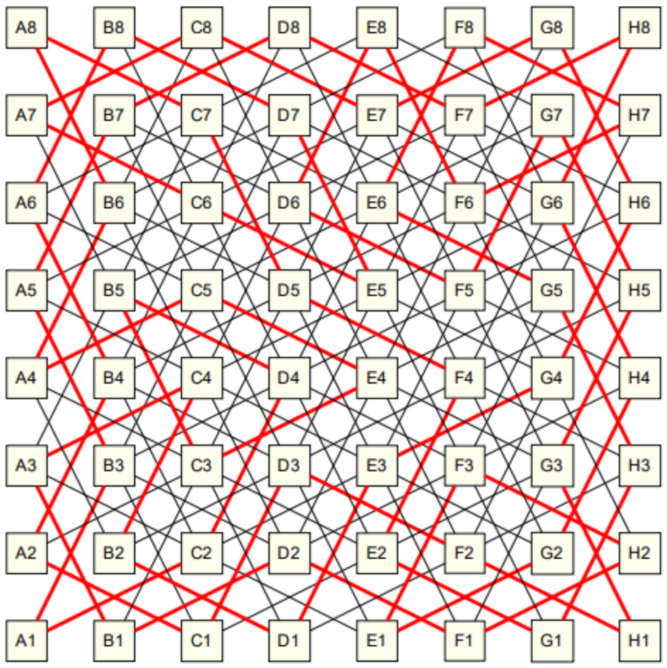

Figure 7: Knight's graph of the 8 by 8 chessboard

APPENDIX B: BONUS POKÉ BALL AWARDS

The number of poké balls awarded for catching a prize Pokémon is dependent on several factors. For achieving a victory in the raid, a base of six balls is earned. Before participating in their first raid, players must choose a team. This team decision may not be changed for that player. There are three options available, Instinct, Valor, or Mystic. Gyms locations are controlled by the team that has Pokémon residing in them. Once controlling a gym six distinct players may place a Pokémon in the gym to "defend" it. During the times when a raid event is not ongoing players from teams not controlling the gym may use their own Pokémon to battle and knock out the Pokémon of the controlling team. When all the defending Pokémon have been defeated, the victorious team may place their Pokémon in the gym. Once a raid event has begun, the team controlling the gym cannot be changed until after the raid. When a player from the controlling team completes a raid, they receive three additional balls. Teams also play an important part of the first damage bonus. Team damage bonuses and individual player damage bonuses are outlined in Table 3. Friendship level also grants bonuses, both for damage and Poké ball rewards. Friendship levels increase by interacting with players often. Table 3 shows how battling with friends affects Poké ball bonuses.

	Number of Bonus Balls		
Bonus	1	2	3
Team Damage*	0% to20%	33% to 49%	≥ 50%
Individual Damage*	0% to 5%	15% to 19%	≥ 50%
Gym Control	–	–	Control gym
Friendship Bonus	With Great Friend	With Ultra Friend	With Best Friend

*Percent of Total Damage

Table 3: Poké ball raid bonuses

APPENDIX C: CALCULATING RAID TIME TO BE USED IN SIMULATION

In order to make a decision on what time to use for the simulated route, 295 raids were timed by the team. The Proc means results showed a mean time of 250.93 seconds and a median of 239 seconds. The skewed distribution, as seen in Figure 8 below, of the times led to using the median for the raid event time in the simulation

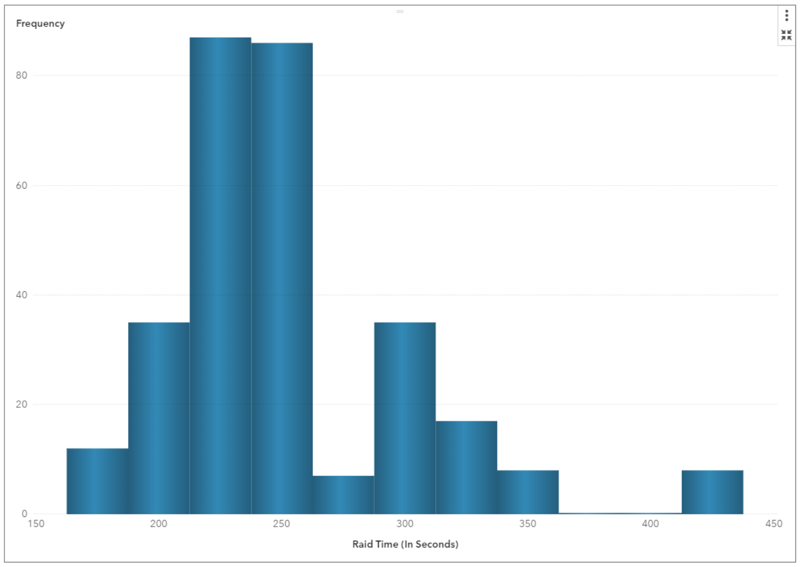

Figure 8: Histogram of Raid Event Times