UCD Smurfit Graduate School of Business



| *Authors:* | | | *Lecturer:* |
|---|---|---|---|
| Eoghan Aherne | 10307151 | MSc B.A Part-time | Dr. James McDermott |
| Ian Brady | 15203975 | MSc B.A Part-time | |
| Conor Reid | 16202630 | MSc B.A Part-time | |


23 April 2017

1. *Describe briefly the real-world phenomenon you wish to model.*

War is the phenomenon that forms the basis for our simulation modelling. We thought it would be an interesting subject due to the observable similarities war and network graphs show, and we believed that it would be possible to simulate the evolution of the world map as different Factions conflicted with each other during the war.

While we found some work done on Network Graph analysis on War, we found very little work relating to the simulation of War through Network Graphs. (Dominic Johnson, 2007) used Networks to analyse Civil War in African countries, and features many of the ideas & properties we hypothesized our network to exhibit such as Population and GDP, more on this later.

If we think about war, it has the properties of infection – as countries are taken over, they can be considered infected, and this infection spreads quicker as more countries are taken over.

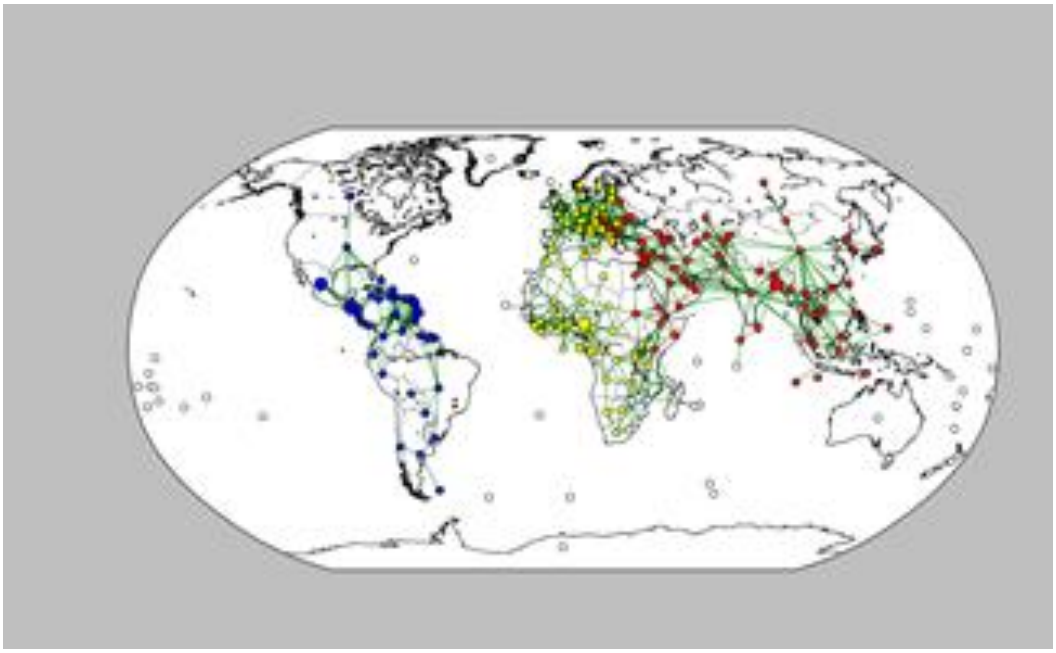
We wanted to model the scenario that a would occur if three Factions declared war on another – not dissimilar to the board game 'Risk'. If a country wishes to take another, they must fight those connected first, gradually spreading out, as a virus or infection would. A country does not 'skip' another one when attacking, rather it attacks countries adjacent to captured ones, that is the neighbours of the node in question.

We liken the chances of infection with the outcome of war. A country will be 'infected' depending on the conflict scores achieved. Once conflict is resolved, the attacking country is either successful and takes over the territory (it is infected) or the defending country successfully repels the attack and remains in the state it was in before the attack (remains uninfected).

If, for example, we consider WWII, Germany initially attacked Poland, in order to arrive in a place where they could attack the Soviet Union from. They had to capture the adjacent country (Poland) in order to progress to the following one, as ultimately, they were interested in capturing the Soviet (not the Polish) land.

The same principles hold true today. While the nature of war has changed since the introduction of manned/unmanned aircraft and more sophisticated marine-based vessels such as aircraft carriers (both of which make it more difficult to model real-world modern warfare), the fundamentals still imply that there must be forces that occupy the ground for it to be considered captured.

If we represent each country with a node, where edges connect it to its nearby neighbours, we can consider the world a network graph. However, if a country has no nearby countries within a certain proximity (e.g. Australia), we will not consider this country to be part of the network. Our representation of the World Network graph can be seen below:



2. *Describe briefly your choice of graph model and how it corresponds to the real-world phenomenon, e.g. your choice of directed versus undirected edges, edge weights, allowing or disallowing self-loops, etc.*

Our choice of graph model was the most straightforward we could use in the simulation and representation of the world geography. It had to be simplistic (due to computational constraints), realistic (as that is the goal of this project) and allow us the flexibility to include the required parameters that would be used to model the graph evolution.

As mentioned, we want to emulate the scenario whereby if three Factions were to go to war, who would be victorious based on several different factors, which will contribute towards an attack score at each timestep. This in turn will determine the outcome of battle and how the simulation unfolds for a pre-defined number of timesteps.

With no readily available data on the adjacency of countries to one another, we decided that the easiest way to create a world graph would be to connect nodes based on the distance between the centres of each country. It was limited by the how dense we could make the graph. That is, we had to decide on a tolerance of how far away nodes should be in on graph before creating an edge between them. Choosing too small a tolerance and there would be too many edges, too large a tolerance and there would be too few edges. We finally settled on a method that accounts for the population and distance between nodes that provided a good balance and populated our graph nicely with edges. We managed to include the majority of the world's countries in our graph, but there were some notable absentees (e.g. Australia) due to how far away it was from the cluster of other nodes on the graph.

We introduce node properties that were readily available online, relevant to determining the battle outcomes and would contribute to our simulations:

*GDP* This is significant as the higher the GDP, the greater the financial might of the country and the greater the country's ability to finance war;

*Population* The army will need to consist of enough people to fight the war. The more people, the higher the likelihood of having a talented, well-trained, motivated army at the country's disposal;

*Geolocation* The latitude and longitude values for the centre of each country to place the node on the map;

*Colour* The colour of each node represents the Faction controlling the country;

Also, involved in the simulations, though not necessarily a property:

*Random Multiplier* In any conflict, there will be elements of luck that will impact both sides. This factor is included to replicate this luck factor.

The properties of our graph were as you might expect – it is sparsely diagonally connected. That is, if we consider a binary matrix, where a 1 represents a connection between two countries, and a 0 otherwise, we end up with a very sparse matrix, giving a sparse graph . This means that the simulation is not computationally expensive and realistic in the way that it is less likely countries would have the ability to attack distant ones, at least initially when they are far away.

It is a **directed graph**, but it is also dynamic, i.e. the graph's edges' direction will change depending on the current Faction. If the Faction owns a territory, the edge connecting it to an adjacent territory is directed – i.e. it can only move that way. Once the next Faction takes its turn, the direction of the edge is reversed (assuming the territory wasn't taken over of course) and the once defending country can now be invaded.

The **edge weights** are represented by the distance between two countries. This is realistic given the likely rise in importance of a connection, the larger the population. This again means that the graph rarely stays static for very long, evolving as the conflicts get resolved each turn.

**Self-loops** are not a feature of this graph or algorithm as the algorithm points and evolves only towards territories that are not currently under the control of the Faction. It can never choose to attack itself or an allied territory, and so we are unconcerned with self-looping.

3. *With the aid of a diagram, describe the simulation rules, e.g. how agents change state and what causes them to send messages.*

Now that we have set up our graph with the required nodes and edges, we can begin to discuss how the simulation runs on the graph.

The simulation starts out with three countries representing the three Factions – *Red*, *Blue* and *Yellow* representing each Faction, with another colour reserved for neutral countries (*White*). These countries are randomly selected, but must have at least a certain Population & GDP before it can be considered a starting point for a Faction. This is where each 'infection' originates from.
For each country under the control of each Faction, the country will attack a neighbouring territory if either (a) it is under the control of rival Faction or;
(b) it is not owned by the attacking Faction or a rival Faction (i.e. it has yet to be taken over by anyone).

We also simulated it such that there could only be one attack per Faction before rotating to the next Faction's attack – more on this later in Section 6.

The order can be simulated or done in a cycle – it shouldn't be a hugely influential. For the sake of simplicity, we chose it to simply take the order of how the nodes were originally added to the graph. Our original data was sorted in terms of GDP and so naturally, the greatest advantage lies with those with the largest GDP's. The model below represents how the conflicts are resolved between Factions:

```
If Population1 > Population2 then
        Conflict_Score = (Population1 / (Population 1 + Population2)) * (GDP1/ (GDP1 +GDP2))
Else:
        Conflict_Score = (Population2 / (Population 1 + Population2)) * (GDP2/ (GDP1 +GDP2))
```

We determine the outcome of a conflict when we compare the above conflict score with 2 random numbers (between 0 and 1):

```
Defence_Score  = (random multiplier * random multiplier)
If Conflict_Score > Defence_Score then
        Add country 2 to Attacking Faction's portfolio
        Remove country 2 from Defending Faction's Portfolio
Else:
        #Do Nothing
```

The conflicts are resolved one at a time, in each time-step. If a Faction is attacking and has a higher conflict score than the Defence score of the territory it is attacking, it is successful and the country being attacked is 'infected' and added to the attacking Faction's portfolio of countries under its control, and is removed from the defending Faction's portfolio. Alternatively, if the attacking Faction has a lower conflict score than the Defence score of the defending country, no change takes place to either Faction portfolio. The pseudocode for the algorithm may be represented as follow:

```
Faction_List ={Red,Yellow,Blue}
For Each Node in Graph:
        Faction = Faction(Node) #Faction in question is the Faction of the attacking node
        For each neighbour of Node:
                If Faction(Neighbour) == Faction:
                Else:
                        If Attack_score > Defence_score:
                                Faction(Neighbour) = Faction
                                Faction_Portfolio.append(Neighbour)
                        Else:
```

The simulation terminates when either the maximum number of time-steps is exceeded or each node is owned by one Faction (i.e. one Faction has conquered the world).

4. *Program your simulation in Python.*

All our code is attached in Zipped folder.

5. *State one or more simulation properties which you wish to investigate and which graph properties you hypothesize they may depend on.*

We may break this down in to three sub-sections, from which we form our hypotheses:

(i) One simulation property that would be of interest investigating would be how the simulation unfolds with different parameters namely, what happens to the convergence time as we when we change the number of nodes and edges in the graph. Given that this makes the graph inherently more complex, we expect this increase convergence time.

(ii) We also wanted to investigate the effect it has if we applied a rule whereby a Faction was allowed only one attack a time before the next Faction. This would theoretically be a fairer simulation and possible more realistic than simple ordering by a country's GDP. We thought that this would result in a quicker convergence towards world domination.

(iii) Finally, we wanted to investigate what the difference would be if there was a cumulative GDP effect vs. using the individual country's GDP every time.

6. *Carry out experiments to measure these properties in different types and sizes of graph (at least one real-world graph, and at least one scalable graph model such as the Erdos-Renyi random graph model, at multiple sizes). Present a table of data showing your results.*

(i) We tested out different variations of our graph, as we varied both the Node count and the Edge Factor, where the Edge Factor refers to the proximity at which nodes must be apart before being added to the graph. We also implemented a condition that would allow in one simulation multiple attacks, and in the next simulation, only single attacks. The results are unsurprising as the table below shows:

Takeovers per turn

| Multiple | | | Single | | |
|---|---|---|---|---|---|
| Node Count | Edge factor | Time | Node Count | Edge factor | Time |
| 245 | 0.5 | 44.4 | 245 | 0.5 | 36.2 |
| 245 | 0.4 | 21.5 | 245 | 0.4 | 17.3 |
| 245 | 0.3 | 6.35 | 245 | 0.3 | 5.85 |
| 245 | 0.2 | 2.42 | 245 | 0.2 | 2.44 |
| 245 | 0.1 | 1.56 | 245 | 0.1 | 1.6 |
| 200 | 0.5 | 30.6 | 200 | 0.5 | 22.6 |
| 200 | 0.4 | 14.1 | 200 | 0.4 | 15 |
| 200 | 0.3 | 4.55 | 200 | 0.3 | 3.66 |
| 200 | 0.2 | 2.05 | 200 | 0.2 | 2.11 |
| 200 | 0.1 | 1.44 | 200 | 0.1 | 1.49 |
| 150 | 0.5 | 17.5 | 150 | 0.5 | 18.6 |
| 150 | 0.4 | 8.77 | 150 | 0.4 | 8.12 |
| 150 | 0.3 | 3.19 | 150 | 0.3 | 3.2 |
| 150 | 0.2 | 1.63 | 150 | 0.2 | 1.8 |
| 150 | 0.1 | 1.31 | 150 | 0.1 | 1.33 |
| 100 | 0.5 | 8.09 | 100 | 0.5 | 7.4 |
| 100 | 0.4 | 4.6 | 100 | 0.4 | 4.17 |
| 100 | 0.3 | 2.1 | 100 | 0.3 | 1.86 |
| 100 | 0.2 | 1.37 | 100 | 0.2 | 1.32 |
| 100 | 0.1 | 1.21 | 100 | 0.1 | 1.22 |

(ii) Contrary to what we expected, when multiple attacks are allowed, it took longer on average, with a positive coefficient in Regression analysis on our output, where Takeovers per Turn is binary, 1 representing Multiple, 0 representing Single:

| | *Coefficients* | *Standard Error* | *t Stat* | *P-value* | *Lower 95%* | *Upper 95%* | *Lower 95.0%* | *Upper 95.0%* |
|---|---|---|---|---|---|---|---|---|
| Intercept | -20.68869636 | 3.927651614 | -5.2674469 | 6.6207E-06 | -28.654343 | -12.72305 | -28.654343 | -12.72305 |
| Takeovers per turn | 1.0735 | 1.918740209 | 0.55948168 | 0.57929756 | -2.8178855 | 4.96488551 | -2.8178855 | 4.96488551 |
| Node Count | 0.072196814 | 0.017686871 | 4.08194381 | 0.00023719 | 0.03632618 | 0.10806745 | 0.03632618 | 0.10806745 |
| Edge factor | 53.36 | 6.783771066 | 7.86583148 | 2.4864E-09 | 39.6018746 | 67.1181254 | 39.6018746 | 67.1181254 |

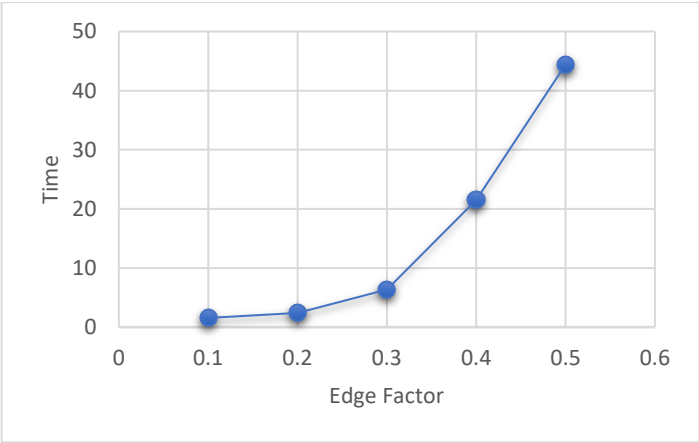
The logic behind this is that when we examined our algorithm, we realised that the loop of conflicts would break as soon as one country was captured under the 'Single' circumstances, as opposed to the 'Multiple' scenario where a full loop must always be executed, and more conflicts must occur before the algorithm can continue. The more expensive loop in Multiple clearly outweighs the likely reduction in timesteps and so takes longer to converge.

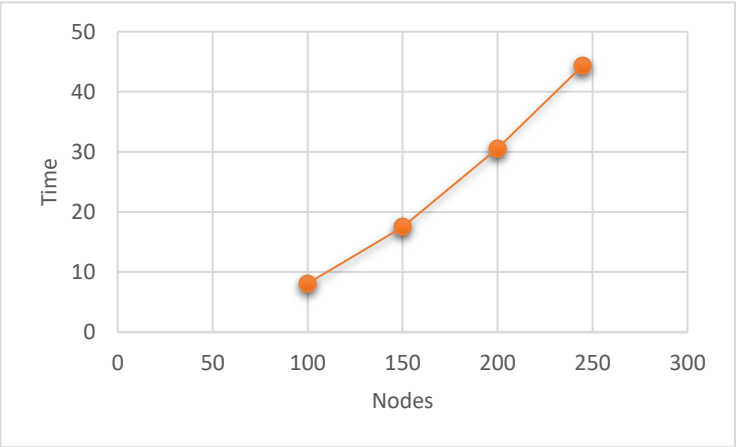
(iii) As we expected, the graph would not converge to one Faction dominating when individual GDP's were used. This is intuitive as eventually, there will come a 'crunch point' where the larger nations continually dominate the smaller ones and are unable to ever take over the larger nations due to the Population and GDP advantages they enjoy, and therefore the outcome of conflict was heavily stacked against the minnows. When this is changes to cumulative GDP per Faction, the combined resources of a Faction are taken in to consideration in the conflict and it becomes much more likely that a Faction can overcome even countries with large GDP's such as Germany or Japan.

7. *State your conclusions, based on your data.*

Our data indicated that this approach was suitable in modelling the simulation of war. The program terminated in a reasonable amount of time with the outcome we had hoped for (i.e. convergence) While there are a variety of other parameters this model forms the basis on how we would develop a more sophisticated model as the successfully converged in our simulations.

The interesting aspect of our model was how sensitive it was to changes in the graph structure. For example, there was a significant reduction in the time of conversion to world domination as the number of edges and nodes we introduced in to the graph reduced. While this was to be expected, there was an exponential increase in the time to conversion as the Edge Factor increases, and a linear increase as the Node count increases:




Given that there are a limited number of possible nodes we could include given the finite nature of the geography of the planet, we are really looking to optimize the Edge Factor. This is well balanced at 0.5, but any increase in this will prove very computationally expensive and questionable given what a larger Edge Factor infers (that more and more of the countries are connected to each other).

We were satisfied with the performance of our algorithm and with the Basemap library, we could watch the graph evolve visually on the world map. We felt that this visualization was key in giving the simulation a real-life feel and makes the it far more visually appealing.

Further work is possible on this. Examples of which may include using adjacent countries to create edges instead of the method we used based on distance in creating nodes between countries. Also, there is certainly many more parameters we could have included in our model that would help more accurately represent a real-life simulation of war. Such parameters may include but are not limited to *Military Spend, Number of fronts*, *Attitude to war*, Experience of Past Wars fought, etc.

Finally, it would be very interesting to see what effect the degree of a node has on how a country taken over. That is, does a country that has a relatively high in/out-degree have strategic value? Do these countries hold the key to victory? In other words, how should a Faction view these territories – should an emphasis be placed on their capture, or should they be avoided?

# References

Dominic Johnson, F. J. (2007). Retrieved from
http://privatewww.essex.ac.uk/~ksg/dscw2007/Johnson.pdf