UNIVERSITY OF SOUTHAMPTON
Faculty of Physical Sciences and Engineering

A Project Report submitted for the award of
Computer Science MEng

Supervisor: Dr Markus Brede
Examiner: Prof Adam Prugel-Bennett

Project Brief

by Nathan Bharkhda, nb7g16, 28404076

February 1, 2018

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING

A Project Report submitted for the award of Computer Science MEng

by Nathan Bharkhda, nb7g16, 28404076

This is the abstract

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

This template was originally from [**?**].

Chapter 2

Background and Literature Review

The Lanchester dynamics were theorises by Fredrick Lanchester in 1916 in order to model attrition rates during a battle in his paper "Aircraft in Warfare, the dawn of the fourth arm". [1] Two independent systems were proposed: Aimed and unaimed models. With unaimed fire, like mortar and artillery bombardment, the rate of loss is proportional to the product of the size of both forces. In my project I will be focused on aimed fire where unaimed fire could be used in support. When a battle consists of aimed fire, the rate of loss of as side is proportional to size of opposing force.

$$\frac{dR}{dt} = k_R B$$

$$\frac{dB}{dt} = k_B R$$

This model contains conserved quantities,

, which allows you to determine the outcome of the engagement from the initial conditions and the attrition factors. The sign of the constant will determine the victor of the battle and using the magnitude and you can find the size of that remaining force.

The model has been adapted and applied to the complexity of asymmetric battles such as in guerrilla warfare, multi-side battles commonly found in civil wars and combined arm combat, the way modern national armies are likely to fight.

It was suggested that to model heterogeneous combat using these dynamics the constants are calculated as "The square of the sum of the square roots of the strengths of its individual units" and size of the force is the sum of each arm. [1]. It has been showed that this assumption that Lanchester made about heterogenous combat is oversimplified and generally do not hold. [2]

The modelling of past battles has been used in order to test the legitimacy of the Lanchester model. The application of Lanchester system to the Ardennes campaign has been studied because of the availability of data for this battle. It was initially found that the campaign fit Lanchester's linear law, however after being reviewed the resulting model was roughly an exponent of 1.5. Some shortcomings of the paper are the oversimplifications of combined arms combat using the outdated oversimplification of heterogenous combat, in addition, with force sizes being so large (1.7 million men in total) and the true battle was the cumulative efforts of several smaller battles. A study into the Battle of Iwo Jima have shown that from the data of troop numbers a Lanchester System can be exist that following the square law. [3] However, the methodology of Engel has been disputed since he used the same data set to estimate constants as he did test. [3] The US Civil War was studied, and it was concluded that on smaller scales the square law fits well when troop sizes are below 15,000 on each side, while with forces larger than that the proportions of those injured increases. In general, most research in comparing real data to the Lanchester Dynamics conclude that on small, individual battle scales, the dynamics can accurately model combat however in larger scales the model falls apart. [4] This may be from the complexity added from advanced tactics and the oversimplification of multiple smaller battles being modelled as a single campaign.

Alterations have been made to the Lanchester dynamics in order to model asymmetric engagements. M. B. Schaffer has developed systems to model guerrilla combat that took place in Vietnam. These models are significantly more complex than Lanchester's original system, taking into consideration more complex factors such as desertion and recruitment rates of the guerrilla force and the effects of morale as a battle takes place.[5] In addition to the addition of extra attrition factors the inclusion of the effects of supporting arms such as artillery results in an incredibly complex model with more involved methods in predicting the outcome.

Modelling modern counter-insurgency operations have some changes compared to their Vietnam era counterparts, for instance these models include the effects of intelligence, the ability to determine the enemy from the civilian population. In addition, the effects of accidental civilian casualties and their role in recruitment is explored as a factor in the model. The inclusion of such factors increases the complexity of the analytic solutions when trying to limit recruitment and having an exponential cost on increasing intelligence.

Multiple factions are common occurrences during civil wars. Modelling these conflicts with an adapted Lanchester system has been used in order to model this. In these situations, the most important point is the time at least one force is destroying. [6] After this it transitions into a regular Lanchester system. There is also the chance that multiple forces are destroyed at the same time resulting in a victor without a transition into a regular Lanchester system. This model technique for multifaction combat has

potential to be easily scales to many more factions. However, in this field there has not been progress in the optimisation of engagement strategy to increase survivability.

The effects of reserve troops and reallocation of forces (aggregating and disaggregation) has been studied by P.K. Davis. [7] This is achieved by modelling having reserve units joining the battle and then having troops on the edges and flanks of the conflict transitioning their focus onto the centre of the battle. The most important factor for a successful battle is the ratios between the time to redeploy troops, the time of the whole battle and the time to deploy reserve troops. For example, if a large proportion of the battle has taken place before the reserve troops are ready to fight, then the likelihood of winning the engagement would be much smaller than compared to if the reserve troops had arrived earlier.

R. K. Colegrove and J. M. Hyde have created a simple heterogenous combat model that follows the original Lanchester dynamics. This model treats every arm of each side as its own force with its own attrition rates, based on the size of the enemy forces. This modelling technique uses Hamiltonian systems, which allows the prediction of the outcome of a battle from initial conditions, like in the original dynamics. This papers scope is limited to the attrition factors being constant along the whole engagement, which generally results in a non-optimal solution for each force.[8]

Optimisation techniques for a single side against a heterogenous enemy has been researched since these are the simplest engagements where a strategy or choice can be made to change the outcome of the battle. Y. Friedman attempted to provide a solution to the optimisation by proving the solution of the homogenous force attacking the arm which will be the most damaged by their attack or the most dangerous arm, one that can do the most damage to the homogenous force. K.Y. Lin and N.J. MacKay combined these two rules to create a general solution which they then proved is optimal in all situations. They suggest that the homogeneous side should wholly focusing fire onto the enemy arm where the product of the attrition factor of the enemy arm attacking the homogeneous side and the attrition factor of the homogenous force against that enemy are is the largest. The proof that this is the mathematically optimal solution to one-to-many engagements was provided and so far, there has not been many challenges to their work.

With the additional complexity found in the heterogenous combat models, finding an analytical solution to our optimisation problem will be very difficult. Therefore, we must use optimisation techniques to find our solution. In order to successfully optimise the problem, we must understand the different techniques that are in use. From the understanding the different techniques, we will be able to discount some methods that are unlikely to work and focus our efforts on techniques with a higher chance of success. The techniques I will focus on are those which are able to avoid getting trapped in local minima.

I intend to test the use of simulated annealing on the model. This is an iterative method which involves picking random points and checking if they are better solutions to the current optimal solution then reducing the spread of the randomness function. This is repeated until the randomness is 0 and a solution is found. This is a useful technique for functions where there may be many local minima since immediate proximity to the optimal solution is not a factor until later in the process.[9]

In addition to simulated annealing I will attempt to use tabu search which unlike other local searches will move to a weaker position if there are no adjacent points with a more optimal solution. This results in tabu search being able to get out of local maxima to more optimal positions. [10]

The final method I will use is stochastic hill climbing because I suspect with the complexity of the model there will be several local maxima so the use of stochastic hill climbing over regular hill climbing would mitigate the risk of getting stuck in local maxima.[11]

By conducting my literature review, I see there is a clear gap in knowledge in this space. The closest two papers to my problem are those about extending the Lanchester Model to (2,2) conflicts by R. K. Colegrove and J. M. Hyde and optimal policy in one against many engagements by K.Y. Lin and N.J. MacKay. With the first paper providing a good framework for my simulation and the second paper providing an analytical solution for a simpler optimisation problem that is very close to my problem. Although these are not directly like my problem, using their research I can verify my simulations are running correctly. My research into the optimisation techniques allows will allow me to correctly implement these and test their effectiveness in the problem.

Chapter 3

Previous Work

Validating the work in the reading with simulations, from progress report

Chapter 4

Design

4.1 Simulation

The simulation function takes in a scenario, which is the initial conditions of the *battle* and an allocation matrix and will return a score, which is the *formula* or the sum of the friend force minus the sum of enemy forces. This function will be used to score allocation matrices. Originally this score was just the sum of the friendly force but the size of enemy force was included in order to allow the functions to score *unwinnable* scenarios where the aim of the allocation is to reduce the losses incurred. The function achieves this by using a while loop that will continue to loop until one of the sides is completely destroyed, or the sum of their force is 0.

4.2 Optimisation Algorithms

For each of these optimisation algorithms I will explain how I intend to implement them, with the help of pseudo code.

4.2.1 Hill Climbing

This should start with a randomly generated allocation matrix. Then the algorithm will enter a for loop. For each iteration a new allocation matrix will be randomly generated. This matrix will then be multiplied by a random small number, this scaled matrix is then added or subtracted to the original allocation matrix. Then the allocation matrix is scored and if this score is greater than the currently selected allocation matrix, will replace the selected allocation matrix. Eventually the allocation matrix will improve however when the program has been running longer the number of iterations required to find a better allocation matrix will increase.

4.2.2 Gradient Descent

Since the allocation matrix is not differentiable with respect to the score, an approximation of the gradient will be calculated by adding a small delta (e.g. 0.0001) and rerunning the simulation. The difference between this new score and original score will be stored in a differential matrix with the same i and j indices of the element that was changed. *INCLUDE PSEUDO CODE TO EXPLAIN* By doing this for each element of the allocation matrix, the differential is approximated. With this approximation of the differential. This differential will then be multipled by the learning rate then added to the allocation matrix, which will then be normalised to fit the constraints. By adding the gradient of the allocation matrix the score should improve since this is the steepest direction, so the direction where the score will increase the most.

4.2.3 Simulated Annealing

Simulated annealing should be implemented in a similar way to hill climbing. The only difference is when the new score is worse than the previous attempt. When a worse allocation matrix is found the a probability threshold is generated based on how worse the new score is and the iteration te score is found. *Insert FORMULA*. A random number will be generated between 0 and 1 and if this number is less than the probability threshold the new lower score will be used in the next iteration.

4.2.4 Genetic Algorithms

The genetic algorithm will start with a population of randomly generated allocation matrices. This initial population will then be scored and sorted from the best scoring to worst scoring from the population. A proportion of the top scoring allocations will be used as a breeding population. From this breeding population 2 parents will be randomly chosen. Crossover takes place by randomly copying columns of the parents' allocation matrix. This also means there would be no need to normalise the matrix as the constraints will not be voided. In order to mutate a random number will be generated between 0 and 1 and if this number is less than the mutation rate a mutation will occur. This mutation will be similar to the *steps* taken in the hill climbing and simulated annealing. This new population is then scored and this repeats for a number of iterations.

4.2.5 Particle Swarm Optimisation

Like a genetic algorithm this algorithm requires a large population to work effectively. A population is made up of *particles*. For each *particle* the population 2 randomly

generated allocation matrices are created. The first one is used as the first allocation matrix for that particle and the second allocation matrix is the particles initial velocity. The *particles* in the initial population are then scored and these scores and allocations are stored as the best position found for each particle. In addition the overall best scoring matrix is saved as the overall best score. After this the next step is calculated by adding the previous step, the difference between the particles allocation that achieved the best score and the current allocation and the difference between the overall best scoring allocation and the particles allocation.

Chapter 5

Implementation

In order to find the most optimal allocations for a given scenario I have attempted to use several techniques. I will discuss my implementation, any trade-offs made, accuracy for speed and any additional tests run to find suitable constants to use.

5.1 Simulations

From previous work on the simplier situations, one on one and one to many engagements, there was a rough template to use, however while one to many engagements require the use of vectors/1d arrays, many to many engagements require matrices to be used. The heavy use of matrix manipulation means that the way the

5.2 Hill Climbing

Implementing what I had designed resulted in significant underperformance. Therefore I decided to alter it. I changed the the way an iteration is counted. The generation of random weights and their addition to the current allocation is repeated until this newly generated allocation scores higher than the previous allocation. I have used a sigmoid function to decreases the scaling of the generated allocation being added to the allocation matrix from 0.01 -¿ 0, so that with more attempts made the size of the *step* taken is smaller so it ensures that the hillclimber doesn't overshoot an improved solution.This implementation has the possibility of being stuck in an infinite loop when the optimal solution has been found. Therefore I have added a time out, which will ensure that the algorithm will eventually end.

5.3 Simulated Annealing

This has been implemented in a similar way to hill climbing. For each temperature there will be a finite number of attempts to find an improved weight. A random weight is generated and scaled, and this is added to the current weight. This temporary weight is then scored, if this new weight scores better than the previous weight, it is stored as the new current weight. This temporary weight can also become the new weight if *Formula* is less than a randomly generated number between 0-1. As the number of iterations increases the size of the the function will decrease so the chances of selecting a worse weight will also decrease.

5.4 Gradient Descent

In this case since the differential of a set of weights is very difficult to analytically create, an approximation is made by increasing the weights in each dimension and a new score is calculated for each increase. *Diagram or formula thing*. This gives the best results out of all of the options, however approximation of the differential becomes more expensive with more complex scenarios.

5.5 Genetic Algorithms

The genetic algorithm has been implemented in as I initially designed it. Cross over is done by a random. I have also attempted a variety of methods to mutate the allocation matrix.

5.6 Particle Swarm Optimisation

Something.

5.7 Multithreading and Parallelism

The running of the simulation is very computationally expensive, on average taking 10-100ms to run. Some algorithms require a significantly large number of times the program will take a day or more to run. A large number of these simulations do not need to be run sequential such as getting the score of members of the population in a genetic algorithm and all the slightly different allocation matrices used to approximate the differential. To improve the performance of these I have parallelised all the parts of

the code that do not need to be run sequentially in order to improve perform. I have used golang's version of thread pools, waitgroups, to parallelise the simulations. These work by adding the number of threads that will run to a counter, when each thread is finished the counter is decremented and a wait function is called, which blocks until the counter returns to 0.

5.8 Graph Drawing

After optimising allocation matrices, the end result is a list of allocation matrices and their scores which is hard to visualise and understand at a glance. Therefore graphs are necessary to easily view the outcomes of the. There are two groups of graphs used: the progression of scores with the number of iterations, used to show the functionality and suitability of the algorithms and the second is graphical view of the allocation matrix. Where each value of the allocation matrix corresponds to the line on the graph. *Add examples?*

5.8.1 Scores

Since python has the best graphing support with matplotlib, I decided to use that.

5.8.2 Network Graphs

Chapter 6

Results and Analysis

Validating the work in the reading with simulations, from progress report

Chapter 7

Evaluation

Validating the work in the reading with simulations, from progress report

Chapter 8

Project Management

Validating the work in the reading with simulations, from progress report

Chapter 9

Future Work

Validating the work in the reading with simulations, from progress report

Chapter 10

Conclusions

It works.

Bibliography

[1] F. Lanchester, *Aircraft in Warfare: The Dawn of the Fourth Arm*. Lanchester Press, 1995. [Online]. Available: https://books.google.co.uk/books?id=nZnfAAAAMAAJ

[2] K. Y. Lin and N. J. Mackay, "The optimal policy for the one-against-many heterogeneous lanchester model," *Operations Research Letters*, 2014.

[3] D. Willard, "Lanchester as force in history," Research Analysis Corp, Tech. Rep., 1962.

[4] H. K. Weiss, "Combat models and historical data: The u.s. civil war," Tech. Rep.

[5] M. B. Schaffer, "Lanchester models of guerrilla engagements," RAND, Tech. Rep., 1967.

[6] M. Kress, J. P. Caulkins, G. Feichtinger, D. Grass, and A. Seidl, "Lanchester model for three-way combat," *European Journal of Operational Research*, 2018.

[7] P. K. Davis, "Aggregation, disaggregation and the 3:1 rule in ground combat," RAND, Tech. Rep., 1995.

[8] R. K. Colegrave and J. M. Hyde, "The lanchester square-law model extended to a (2,2) conflict," *IMA Journal of Applied Mathematics (Institute of Mathematics and Its Applications)*, 1993.

[9] D. Connolly, "General purpose simulated annealing," *Journal of the Operational Research Society*, 1992.

[10] M. Gendreau, *An Introduction to Tabu Search*. Boston, MA: Springer US, 2003, pp. 37–54. [Online]. Available: https://doi.org/10.1007/0-306-48056-5_2

[11] J.-H. Wu, R. Kalyanam, and R. Givan, "Stochastic enforced hill-climbing," in *Proceedings of the Eighteenth International Conference on International Conference on Automated Planning and Scheduling*, ser. ICAPS'08. AAAI Press, 2008, pp. 396–403. [Online]. Available: http://dl.acm.org/citation.cfm?id=3037281.3037331

Appendix A: Photos

This is an appendix

Appendix B: Code Listings

This is an appendix