FitForge Allocation System Proposal

T039

OLIVER PINEL – ALEXANDER KEARNEY – RAPHAEL FAHEY – CALUM INGRAM

Table of Contents

Introduction

Over the past semester, our team has been investigating how to best develop a new system to aid in allocating teams to projects for the Unit Coordinator and Teaching Staff of the IFB398 unit. In order to do this, the considerations and requirements of the system were determined. By using them, the best way for the software to operate and, consequently, the key components could be identified. Through investigations into possible methods for developing each component, potential implementations were considered and compared, allowing us to determine the most optimal options for each. Following from this, we have developed a plan for how we intend to implement the system, and how it works. This report will detail our work through these sections, describing our options, considerations and decisions in depth, as well as our plan for the further development, functional implementation and testing in the next semester.

Purpose

The goal of this project is to develop software to aid in the allocation of teams to projects based on both numerical data and external intangible factors. As such, we must first consider the original method of allocating teams to projects, as to properly identify how it can be improved.

The original task involved considering m teams and n projects, and allocating a project to each team, given a set of criteria. To do this, each team and project had characteristic data, which could be defined either in numerical exact values or intangible factors beyond the description of simple numerical and computational consideration. Each team had up to four team members, each who had completed some units previously with some level of academic result. In addition, each team has areas that they are more likely to be proficient in, based on their major. Each project also has factors that are be considered. These include the level of impact of the project, such as its difficulty and the importance it is completed. Finally, each team can indicate some form of preference for each project.

The details of each numerical consideration between team and project are beyond the scope of the investigation. Thus, for this report and software development, three numerical values will be considered, and here we shall define some notation to be used through the report.

First, we define a variable b_{ij}. This variable b is the benefit function, describing the numerical consideration of the allocation for team i to project j. As explained, this variable b is a combination of the three numerical characteristics as follows.

$$b_{ij} = b_{impact}(team, project) \times \left(b_{capability}(team, project) + b_{preference}(team, project)\right)$$
$$i = \#team, j = \#project$$

This allows for the formation of an $m \times n$ matrix describing the numerical considerations for every team for every project.

In addition, a second variable can be defined.

$$x_{ij} \in [0,1]$$

This variable x describes whether team i is allocated to project j. If the value is 1, then they are allocated, otherwise it remains as zero and they're not allocated. This allows for the formation of a second matrix the same size as b to describe the allocations proposed.

From this, we have defined numerical values to describe how a team would match to a project. In addition to this are other intangible considerations, that must be made by a human. The detail of these is again beyond the scope of this report, and it will suffice to say that each allocation involves a numerical consideration, weighted against intangible considerations.

Using this system, the overall goal of the task can be achieved: to allocate each team exactly one project, and to allocate up to the maximum number of teams that each project can accept to maximise the overall satisfaction with the allocations.

Due to the complexity not just of the data, but how spread out it is, when scaling the project to the size expected to be done, of approximately 100 teams and 100 projects, the allocation becomes significantly difficult, taking a large amount of time, and threatening accuracy, particularly as it is done purely by a human. Therefore, we will be investigating ways to improve this.

Requirements and Considerations

By considering the problem at hand, there are several main criteria that can be identified as the major problems with the current method, leading us to be able to determine what the requirements would look like for an improved system.

One consideration is the time taken to perform allocations. Due to the complexity, size and spread of the data, the current implementation is tedious. As such, it would be a significant improvement to condense the data as much as possible into one location. By reducing the amount of time spent manually considering a large spread of data, and highlighting the best options, the time taken by the human using the software could be significantly reduced.

In addition, a computer given enough time is capable of determining some form of a mathematical optimal choice far more accurately than humans can. In the current system, the time taken to perform the allocations is traded for accuracy, and although longer time spent generally should improve the set of allocations determined, this is not always guaranteed. Given the definition of accuracy being "as close to the most optimal set of allocations of teams to projects that maximises satisfaction", at a large scale this is both very important and very difficult to optimally do. As such, one of the goals of the new system will be to improve the speed, both in terms of collecting data, and producing allocations. The system developed should be quick to provide an allocation, and just as fast for the user to retrieve the relevant data. This will be one of the foundational concepts that the system will be intending to improve and will be reflected in the choices for the development.

As was just mentioned, another significant factor, arguably the most important, is the accuracy of the system. There is no benefit of the system being very quick to produce allocations, if the allocations are very suboptimal. While it is unlikely that a perfect solution could be found for every team and project, striving the reach that is the central facet of this task. As such, any system developed must have a significant focus on producing a result as accurate as possible, through a combination of some computational method and human input. When considering potential options to develop the system, the ability to produce accurate allocations will be a fundamental component considered.

Finally, the human element of the process cannot be ignored. Whilst the intangible considerations play a factor, they are not the only reason that a human element must be involved. The allocations produced during this process will have a significant impact on the lives of the students and industry partners over the following year. For that reason, it is essential that the choices made be properly justified by a human. It is considered unfair for a purely numerical computer algorithm to produce a best fit and force the consequences of potentially bad allocations onto the students. The allocation must be approved by a human, meaning any system developed needs some form of a user interface to view an allocation, to view the numerical data and intangible as much as possible, so that the user can make an informed and justified decision.

General Proposal

Based on the considerations defined, for speed, accuracy and human impact, we propose a system that is developed in the form of two specific components. By considering the size and details of the data available that must be considered, by condensing the purely numerical values of compatibility, fit and impact into a table of b values, a purely algorithmic allocation based on that can be produced. Since it is difficult for the intangible considerations to be factored into a computer algorithm, the best option is to ignore them, and produce a purely mathematically optimal solution to the allocation problem. This proposed allocation will be as fair as possible in the sense that it is the objectively best allocation based purely on b values.

From there, we suggest the implementation of some form of user interface. This will be the second component to the system developed. Once a purely optimal allocation is produced, there must be some way to view, consider and either approve or make changes based on the intangible values. The user of the system must be able to view each allocation in the proposed set based on their three values, and in addition easily be able to retrieve the intangible data as best as possible. From there, they can for each allocation that seems problematic indicate not to allocate that, and then allow the system to produce a new allocation.

To summarise, we intend to develop the system with the following workflow. Initially, the user will upload at least the numerical data, and potentially some metadata for each team and project, being the intangible considerations. From there, an algorithm will generate an initial allocation, that is purely algorithmically based to maximise the numerical data. Once this is done, the user will have a chance to sort the allocations and identify undesirable suggestions. Then, a new

allocation set can be generated by the algorithm, based on the previously identified changes. This should be repeated until an allocation has been produced to the satisfaction of the user.

Algorithm

As explained, one of the two components of the system we will develop is the algorithm. The purpose of this tool will be to take the numerical data and the user decisions based on the intangible considerations and produce the best fit possible of a set of allocations from teams to projects. The goal is to determine a specific method of allocating that can abstractly determine the best fit without any external considerations, that fits the criteria we determine. Due to the importance of this section being accurate, we will endeavour to investigate already-defined algorithms, that have been tested and optimised by other software engineers. While we could potentially develop our own specific algorithm to solve this problem, it would likely be badly implemented and inefficient. Exploring pre-designed algorithms will improve our chances of implementing the best solution for the system. This section shall explore the requirements, considerations and criteria that were derived to use when considering potential algorithms, the options we investigated, and why we have determined which ones to use.

Requirements, Considerations and Criteria

Before investigating different options for well-defined algorithms, we must define the requirements that will be considered when investigating them, and the criteria with which we will be comparing them. As was identified, the goal of the algorithm is to be accurate and fast. This means that the potential algorithm should be capable of accepting numerical data in the expected form of capability, preference and impact, and using this data be able to determine the best set of allocation options.

The major considerations when investigating according to this goal is that the algorithm is fast, accurate, scales well, and is also extendable. As the input data is possible to shift in the future, an important consideration is how well the algorithm can be adapted and used later. Ideally, an algorithm can be chosen that will allow for future modifications, without needing to be replaced. As such, a generalised algorithm is preferred over a specifically accurate one, yet the algorithm must still remain applicable to the problem without being too general that it would require significant modification to work. The algorithm should return some matrix or set of x_{ij} being 1 for team i being allocated to project j as the proposed algorithmic allocation.

Finally, the chosen algorithm should be simple in complexity and size. It is undesirable to implement a large and complex algorithm when a smaller one would suffice. This is by no means a fixed decision, but a large algorithm significantly increases the complexity required to properly implement it, and limits the possibilities when designing the overall software.

Investigated Options

Once we had determined our requirements and criteria, broad research was done into potential options. We worked simply by investigating the topic “allocating methods and algorithms” and collected notes on potential options. The goal at this stage was to identify a wide range of options, so that our investigation would likely identify the best possible choice to use. As such, we collated our notes and identified 12 different concepts and methods that are well known and defined in types of algorithms involving allocations. From there, we performed a deeper investigation into each one, and categorised them into three distinct groups.

Dismissed Options

Market Mechanism
While this option was mostly considered humorously, and it did not eventuate as a truly viable option, considering the process of allocating teams to project in the form of a market-based mechanism was surprisingly realistic. As each team had a numerical value representing their fit for a project, it was possible to visualise the problem in terms of an auction or trading centre, where teams could use their score as a currency to obtain a project they wanted. By trading and bidding, a simulation of a market could have been used to perform the allocation. While this is a viable method, it is an abstracted version of the allocation problem, intended for more complex and sprawling problems with a wider variety of variables (Condorelli, 2013). As the actual allocation of such a market would involve some other algorithm, the purpose of recontextualising the problem became pointless, and although entertaining as an idea, it was discarded as irrelevantly overcomplicating a simple situation.

Collaborative Filtering Technique
The collaborative filtering technique was considered an option based on its use as a recommender system in many sites in use today. It was initially thought that this could be extended to this system, however that was found to not be the case. A collaborative filtering is a method of determining determine what a person might enjoy based on their previous preferences and the preferences of others who have similar tastes. We intended to consider this on the basis of taking the preferences for teams and allocating them to projects with similar preferences. This would provide an option as an extension on the current data structure. While this may be worth investigating in the future as an addition to the implemented algorithm if the preferencing system is changed, the nature of collaborative filtering means it requires a high sample size to have any degree of accuracy (Mustafa, Ibrahim, Ahmed, & Abdullah, 2017). This, in addition to the modifications that would be required to utilise this successfully, as the filtering technique is different than what is needed for this system, meant it was discarded from consideration.

Pareto Efficiency Algorithm
Pareto Efficiency is a situation wherein which no option exists that makes one individual better without making another worse. This was a concept we explored when investigating methods of allocating, as by nature of improving an allocation set during an algorithm, a change to one team’s allocation may make another team worse (Rasure & Logan, 2024). It describes an iterative process of improvements, making choices that would not cause any other team to reduce in satisfaction. While this in concept sounds promising, the topic was too generalised, designed for producing models about optimal choices, called production probability frontiers that indicate

why a choice is good, originating from finance, and while it would conceptually apply to this situation, more optimised algorithms that are explored further on, implement this same logic in a situation specific to this problem. There is potential for implementing this production probability graph as a demonstrable way of justifying choices as a further extension in the future.

Ant Colony Optimisation

The Ant Colony Optimisation Algorithm was a unique method of allocating. It proposed a hybrid combination of a genetic algorithm. It takes advantage of the random improvements of genetic algorithms in subcomponents styles as “ants”, significantly reducing the time taken for each genetic algorithm to improve due to the reduced size and leverages the multiple “ants” to build multiple combinations through a hierarchical structure. Without going into details, it utilises the nature of ants to trace pheromones as they move to reduce the inefficient variations in the probabilistic choices (Salami, 2009). While this algorithm is a significant improvement, particularly for this relatively simple problem, the nature of genetic algorithms is still not especially relevant to this problem. It suffers from the same issues that the genetic algorithm did, albeit reduced due to the optimised hierarchical algorithmic nature. While it is conceptually interesting, the algorithm offers nothing particularly special, and this was not worth considering compared to other algorithms, beyond the uniqueness factor.

Viable Options

Genetic Algorithm

One option seriously considered was implementing some form of a genetic algorithm. These algorithms create a solution in the form of a chromosome-like data structure, which in this case would be an allocation set, and apply recombination operations based on the score of each until the ultimate ‘evolution’ is found (Mathew, 2021).

This option would certainly arrive at an optimal solution and may even find unique sets of allocations that a standard algorithm wouldn’t reach. In addition, due to the customisable nature of the chromosome-like structure, and the method of the algorithm to naturally improve through random changes no matter the data, this implementation would certainly allow for future changes to the input data (Alhijawi & Awajan, 2023).

However, due to the random nature that is employed to replicate genetic changes as a growth optimiser model, the time taken could not be reliably managed. In addition, due to the design of the algorithm to model genetic change, the algorithm is much more reliable when run in tandem with multiple simulations and data, increasing both the time and complexity of such an implementation. While it certainly has potential, and can be further investigated if required, due to the better options available it has been rejected.

Reinforcement Learning – Machine learning

One of the first options we considered was an implementation of machine learning. Machine learning is a method of taking in data and over time creating. A decision tree that can solve complex problems in ways that humans cannot easily do. This happens through the training of data, and can incorporate a wide range of data type, potentially even the intangible data, which was what first made it appeal (Bao, Peng, Wu, & Li, 2018); machine learning could allocate everything at once.

Despite the appealing nature, machine learning was rejected as an option for two reasons. Firstly, the amount of training data required was simply infeasible. In order to develop a well-balanced neural network, or some form of network generated by a machine intelligence algorithm, training data is needed to "teach" the algorithm. This allows it to know how to improve the decision tree and output better choices (Brownlee, 2019). For instance, the data from previous years b values and intangibles would be used, and the set that was chosen would be outputted, and the algorithm would use that to generate more. Unfortunately, as previous unit's data is not readily available, it is not possible to train a machine learning algorithm. In addition, machine learning algorithms are quite computationally heavy, something we intend to avoid, making it even more of an undesirable choice (Joloudari, Nodehi, Alizadehsani, & Mojrian, 2022). In the future, it may be worth revisiting, but for now it has been rejected.

Heuristic Search

A heuristic algorithm is a problem-solving approach that employs practical methods and shortcuts to produce solutions that may not be optimal but are sufficient for reaching immediate goals. These algorithms are designed to efficiently handle complex problems where traditional methods are too slow or fail to find any solution. They are commonly used in scenarios requiring quick, approximate solutions, such as in search engines, AI, and optimization problems (Gao, 2021). An implementation of this can be used to solve this problem, as due to the size a unique and different method to a greedy iterative algorithm could prove more effective, as seen in the time taken for the current implementation.

Despite the benefits of this algorithm for this specific problem, the complexity of the options vastly outweighs the needs. Like the neural network, this implementation would be overkill for a problem, that while is large in data, is straightforward with only a few different perspectives of variables to consider (Hansen, 2001). In addition, while the algorithms are optimised, our ability to implement algorithms of this complexity is in doubt, and as such it remains as a reserve, should we need to do further investigation.

Hungarian Algorithm

The Hungarian algorithm is a combinatorial optimization method used to solve the assignment problem, which involves finding the optimal way to assign tasks to agents at minimum cost. It operates by constructing a cost matrix and repeatedly performing row and column reductions, as well as augmenting paths, to minimize the total assignment cost (Moore, Landman, & Khim, 2024). This would work for our problem as an implementation to find the best method, particularly since it comes equipped with expecting a data structure in the form we are working with. It would be a good choice, since an implementation is small, and quite accurate.

Despite this though, the Hungarian Algorithm is well known for its inefficiency, which is generally accepted as the trade-off for its accuracy. Being at best optimised in time $\mathcal{O}(n^3)$, any implementation would be potentially quite slow (Columbia, 2012). For this reason, it has been set aside for hopefully faster algorithms to be implemented. As the speed is the only concern, should the other algorithms fail, this is a likely backup to be investigated.

Options for Further Investigation

Gale-Shapely

The Gale-Shapely algorithm is a well-known algorithm designed for the solution of the stable marriage problem; that is, an algorithm to allocate one group to another based on preferences that both groups have (Osipenko, 2019). For the purposes of our task, where each team had a numerical preference for each project, this was an ideal algorithm to implement, as it was designed for this specific problem, and had decades of optimisations and improvements.

The algorithm works through an implementation of a bipartite graph, collecting the two groups and joining each element from the teams' nodes to the project's nodes, with each edge having a weighting. From there, it iterates over each team and finds them better matches until the allocations can't be improved, running at worst in an $\mathcal{O}(n^2)$ time (COHN & RHODES).

This algorithm is lightweight, and configurable for different data inputs, but most importantly is very adaptable to our current situation. In addition, it is computationally fast, taking at most n^2 for n teams, which is a benefit over the Hungarian algorithm. For these reasons, it was investigated further, with an attempt at implementation made.

Integer Linear Programming

Integer linear programming is a different option to many of the previous algorithms, in the sense that it is a mathematical optimisation method, as opposed to an algorithmic improvement. In this setup, the teams and projects are considered variables, and by expressing the constraints of the model, a linear system can be derived (Oesper, 2024). For instance, given x_{ij} represents the selection of team i for project j, an equation can be formed for team i:

$$x_{i1} + x_{i2} + \cdots + x_{in} = 1$$

Which mathematically defines that team i can be allocated to exactly one team. By forming a system for every team and problem in this manner, the following maximisation problem can be solved:

$$maximise(\sum_{i=1}^{m}\sum_{j=1}^{n} b_{ij}x_{ij})$$

Given that b_{ij} are the mathematical data as defined earlier, linear algebra can be used to maximise the selections of x_{ij}.

This method shows great potential due to how configurable each constrain of the problem is. It can be modified to handle data and is additionally a lightweight program storage-wise and computationally. As a result of optimisations to the mathematical problem $Ax = b$ for many years, Integer Linear Programming has been improved as a solution due to its common real-life usage (Graves, 2023). Finally, as the algorithm is purely mathematical based, it will guarantee the optimal allocation, based on the constraints provided. While some algorithms may provide variations due to iterations and randomness, this algorithm guarantees the same "perfect" allocation each time. For these reasons, it has been chosen for a tested implementation.

Simulated Annealing
Simulated Annealing leverages a probabilistic-like behaviour to determine the optimal solution to a problem, wherein many “good” solutions exist. As a concept, it describes movement from an initial solution to a better one in the form of a gradient slope, and probabilistically choses the most likely direction for improvement and moves towards that (Auton Technologies, 2004).

This concept of the best of good options was particularly interesting for this situation, wherein many allocation sets would be considered viable, but identifying one over the others is complex. One flaw with algorithmic problems is they can end up trapped in a state that is impossible to improve without reducing another, as the Pareto Efficiency generalised (Bertsimas & Tsitsiklis, 1993). This algorithm aims to avoid that through a rather complex mathematical design, that was beyond our direct understanding, and we required example implementations to follow how it worked.

The algorithm is quite complex, and it is not clear how well it can scale in size due to the iteration. Despite this, the uniqueness of the algorithm, in addition to its ability to reliably produce accurate results makes it a valid candidate to explore further. While it is not clear how well it fits the speed requirement, the flexibility and accuracy give it quite a bit of potential.

Network Flow Algorithm
A Network Flow Algorithm is similar in concept to the Gale-Shapely in the sense that it involves graph theory but differs from there. After implementing a connected graph with weighted edges connecting teams and projects, a network flow attempts to find the best path through the system such that all the edges are connected (II, 2006). Through optimised graph theory it is possible to efficiently solve this and determine the best path to traverse.

The algorithm uses the data provided as weights and can path through the options to find the best allocation system. This suffers from the potential problems as other algorithms, in the sense that by following the wrong path it can produce a suboptimal allocation, but due to the simplistic nature of graphs, it is both small and fast, ideally not requiring too much computational power (Network Flows, 2008). In addition, due to the simplicity of graphs, it should be possible to modify the data structure as needed. Due to the promise of this algorithm, it will be implemented for further testing.

Recommended Implementation

Once we had identified the algorithms with most promise, we proceeded to create implementations, in order to compare them, and determine how well they matched the requirements in practice. By generating random data according to the expected input data, we could then measure their efficiency and accuracy.

Of the four, the network flow was the first to not work, as we struggled to create a working implementation. As a result, we could not properly test it. For the time being it has been set aside, to come back to if we need to revisit other options.

Next, the Simulated Annealing was performing very well on small scale problems, with a high accuracy matching the numerical accuracy of Integer Linear Programming, and speed matching the Gale-Shapely Algorithm. However, once it was scaled up to the test size of 100x100, as expected for the real-life use case, it began to slow significantly, and became very inefficient, and was thus not useful for the task.

Gale-Shapely and Integer Linear programming both performed excellently when tested on different data sets, both running quickly and with high accuracy. When compared with a score, Gale-Shapely was found to be consistently slightly faster in computation time by several seconds, while the Integer Linear Programming was consistently producing the optimal allocation set even as the Gale-Shapely produced some slightly suboptimal sets.

As such, it is the recommendation of the report that we perform a dual implementation of Integer Linear Programming and Gale-Shapely. Running these in parallel and using both outputs as needed should provide quick and accurate results, as well as more flexibility for the user. As will be explained further on, during the testing we will gain a clearer understanding of which algorithm is optimal to be used, or whether some previously disregarded algorithms, such as the Hungarian, should be revisited.

User Interface

The second component of the system is the user interface. While the overall environment and development of the design is not essential, the interface must be capable to ensuring the software meets the requirements and criteria, allowing the user fine control in an efficient and timely manner, to produce changes to the allocation output. This section will highlight the potential options for overall user interface design, their considerations and requirements, and the final recommendation.

Requirements, Considerations and Criteria

As explained previously, it is essential that the software be easily interactive for the user. But most importantly, it needs to be able to pass from unit coordinator to unit coordinator, be used on potentially any system, possibly from anywhere. The user interface must be accessible as possible, and customisable as much as possible, in order to give the best opportunity to develop something that could be used for years to come. As such, there are three main categories for ways to host some form of user interface that will be explored.

Investigated Options

Desktop App

The most obvious solution initially is to develop a desktop app. This, however, is objectively the worst choice. Due to the uncertain nature of the device being used for allocating teams to projects, the desktop app would have to work for every possible device. In addition, when updates for the operating system are released, it is possible that the device would no longer

support the app. Developing a desktop app is very OS locked, and as such would be difficult to extend or modify in the future, as designing the specific features of the user interface will certainly be subject to change as development and testing progresses. As such, a desktop app is should be avoided if possible.

Google Scripts Extension on Google Sheets

Since the current system was implemented with a combination of google sheets and R, and in particular the data was majorly collated within google sheets, it would be possible to implement the system as a google extension, designing some form of interface as an extension to sheets that can perform an allocation and display the output entirely through the system.

After some research into google scripts and the functionality it would provide, it was found that while this was possible, it would not be a desirable method to go. Google scripts are designed for simple tasks, and while outputting into a sheet is possible, google scripts are not designed for heavy computations, likely what would be required for our algorithm. Again, this is possible to be worked around by hosting a program with google cloud, and accessing it through an API, however this would potentially cost money, and bring about concerns in security and privacy of data that is restricted. In addition, this would be just as difficult to pass between teaching teams as any other version, so the only benefit would be in the centralisation of the data into the google environment. While a google scripts extension to enable the work of a google sheets is possible, it is not the best option available.

Web Application

A web application would involve developing a user interface in Javascript through NodeJS, and implementing a website which can load, calculate and represent the data as needed. This would involve using some frontend library that can interface with a backend algorithm implementation to read and store the data, and then represent it. In addition, a web application is capable of being extended in many ways. It can be hosted on a server and accessed locally or through the internet, implemented such that it can be accessed in the browser of only the laptop that has it, or even through the use of libraries such as electron be developed into a near-universal desktop app. For the purposes of this project, while we still determine specifics of how we wish to implement the overall program, a web application offers the freedom to make these choices later, and still develop a working prototype.

React is the most popular frontend library for Javascript, due to its simplicity, and it is a framework we as a team are familiar in. Due to its nature as a component driven framework, it is very easy to design a user interface as needed, to best fit the design that allows the user the most customisability. Combining this with Javascript's wide range of libraries available, it is possible to implement all the algorithms considered above in it. As a result, the project could easily be done in this nature, leaving us free to consider final implementations while developing the prototype separately.

Recommended Implementation

As such, it is the recommendation of this report that a Javascript implementation be designed using the React library to create a user interface that fits the situation. It is the most general concept, and as we become clearer on the goals and ideas of how we wish the software to be used, we won't be hindered by our choices made so far. This will allow us to meet the criteria of the application being quick and easy to use and giving the user absolute control over the allocation options as much as possible.

Proposed Solution

Technical Summary

Based on the decisions outlined in the previous sections, our goal for the application and it's development can now be formed and outlined in detail. We intend to develop a two-factor solution to the problem, by implementing an algorithm and user interface that allows the user to accurately and quickly create an allocation set of teams to projects, and then make changes until they are satisfied with it. In order to do this, we will develop a combination of algorithms, implementing a version of the Gale-Shapely algorithm, as well as an Integer Linear Programming implementation, so that both can be run on the data provided. From there, a user interface will be developed with react, and hosted on the local device. This interface will allow the user to upload the student benefit values. From there, the system will run the Integer Linear Programming algorithm, producing an initial allocation set.

With the allocation set, the React app will render it in a table view, showing each parameter, and the intangible data if possible, so that the user can see the proposed allocation set in detail. They can sort through it and identify bad allocations and ones they would like to keep. Following this, a new allocation can be produced. During this time, which algorithm being used can be changed, although both should be run just to visualise the "score" of the set. Previous allocation sets should be able to be revisited, and allocations fixed or rejected should be able to be visited again. This process should repeat until the allocation set is satisfactory, not until every allocation is made. During our experimenting with smaller data sets, by forcing some allocations, it significantly lowered the potential range for others, and the eventual set was suboptimal. It should be noted, then that the fixed allocation should be avoided, and bad allocation preferred as an indicator. Finally, a final set of allocations should be produced, and exported into some useable form.

Plan for the Upcoming Semester

Based on the structure above, we plan to implement this design into the creation of the software in three distinct stages, each with their own features and purpose.

During the second week of next semester, the next unit will begin, and the teams formed from the students will be allocated to their projects. This is the most important moment in the project,

as the real-life data will allow for the project to be properly tested. In addition, we will develop a questionnaire to record feedback and determine how the system functioned, and what could be improved. As such, by week 2 of next semester we aim to have:

- A working prototype that can:
    - Read in b value data
    - Indicate which projects can take more than 1 team, and how many teams
    - Potentially read in other metadata
    - Process this data into the right form
    - Use Gale-Shapely and IPL to produce an allocation from the data
    - View the allocation in readable form
        - Team – Project – bvalue – other info?
    - Sort by allocation satisfaction
    - Indicate that a team should always be allocated that project
    - Indicate that a team should never be allocated to that project
    - View previous allocations
    - View defined rules about team/allocations
    - Save the allocation to a spreadsheet output

Once the testing phase is done on the data, we will have a better idea as to how the current idea for the system will work, and whether we can progress or return to other ideas that have been presented as rejected options. From there, we will begin work on refining the prototype, into a fully functional version. We intend to have a working version that could be utilised next year if we abandoned the project by the beginning of the mid-semester break. This version will have the singular goal to functionally meet all the initial requirements considered, be able to be passed on to the next teaching team if required and require little maintenance and explanation in how it should work. We intend for this iteration to be our completed version 1.0 of the system. To test this, we intend to perform one more test, and again record feedback as to how it can be improved, and any changes that the user feels should be made.

Following the mid-semester break we will focus on the final remaining ideas that could be incorporated into the project but are not essential. Ideas such as server hosting, login systems, data storage, more complex algorithms. There are many features that have been considered that are not directly necessary but would potentially improve the system. We intend to use the remaining time developing any user wants, now that the needs would be complete.

While we intended to have a full release, plan prepared to accompany this report, as the testing will mark a major point in the understanding of the success of the software. As such, this general guide shall be followed, and then refined once the testing phase returns details on which direction to progress in.

Works Cited

Alhijawi, B., & Awajan, A. (2023). Genetic algorithms: theory, genetic operators, solutions, and applications. *Evolutionary Intelligence*.

Auton Technologies. (2004). *Simulated Annealing*. Retrieved from Auton Technologies: https://www.cs.cmu.edu/afs/cs.cmu.edu/project/learn-43/lib/photoz/.g/web/glossary/anneal.html

Bao, Y., Peng, Y., Wu, C., & Li, Z. (2018). *Online Job Scheduling in Distributed Machine Learning Clusters.* Honolulu: IEEE.

Bertsimas, D., & Tsitsiklis, J. (1993). Simulated Annealing. *Statist. Sci.*, 10-15.

Brownlee, J. (2019, May 23). *How Much Training Data is Required for Machine Learning?* . Retrieved from Machine Learning Mastery: https://machinelearningmastery.com/much-training-data-required-machine-learning/

COHN, S., & RHODES, A. (n.d.). *AN EXPLORATION OF STABLE MATCHING AND THE GALE-SHAPLEY ALGORITHM.* Amherst: University of Massachusetts; Department of Mathematics.

Columbia. (2012). *The Hungarian Algorithm.* Columbia: Columbia University.

Condorelli, D. (2013). Market and non-market mechanisms for the optimal allocation of scarce resources. *Games and Economic Behavior*.

Gao, A. (2021). *Heuristic Search.* Toronto: University of Toronto.

Gharaibeh, A., Ali, M., Abu-Hammour, Z., & Al Saaideh, M. (2021). Improving Genetic Algorithms for Optimal Land-Use Allocation. *Urban Planning and Development*.

Graves, S. (2023). *INTRODUCTION TO INTEGER LINEAR PROGRAMMING WAREHOUSE LOCATION.* MIT.

Hansen, E. A. (2001). LAO∗: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence*, 35-62.

II, L. G. (2006). Network flow analysis algorithms. *Ecological Modelling*, 586-600.

Joloudari, J. H., Nodehi, I., Alizadehsani, R., & Mojrian, S. (2022). *Resource allocation optimization using artificial intelligence methods in various computing paradigms: A Review.* ResearchGate.

Mathew, T. V. (2021). *Genetic Algorithm.* Mumbai: India Institute of Technology Bombay.

Moore, K., Landman, N., & Khim, J. (2024). *Hungarian Maximum Matching Algorithm* . Retrieved from Brilliant: https://brilliant.org/wiki/hungarian-matching/

Mustafa, N., Ibrahim, A. O., Ahmed, A., & Abdullah, A. (2017). *Collaborative filtering: Techniques and applications.* Khartoum: IEEE.

Network Flows. (2008). In B. V. Roy, & K. Mason, *Introduction to Optimization* (pp. 107-118). Stamford: Stamford.

Oesper, L. (2024). *Integer Linear Programming: What? Why? How?* Retrieved from Carleton College: https://www.cs.carleton.edu/cs_comps/2324/integerLinearPrograming/index.php

Osipenko, A. (2019, May 27). *Gale–Shapley algorithm simply explained*. Retrieved from TowardsDataScience: https://towardsdatascience.com/gale-shapley-algorithm-simply-explained-caa344e643c2

Rasure, E., & Logan, M. (2024, February 28). *Pareto Efficiency Examples and Production Possibility Frontier*. Retrieved from Investopedia: https://www.investopedia.com/terms/p/pareto-efficiency.asp

Salami, N. M. (2009). Ant Colony Optimization Algorithm. *UbiCC*, 823-826.