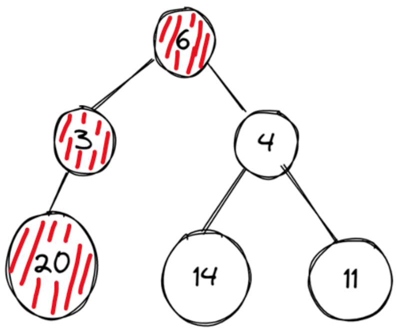
THE MINIMIZATION OF GREEDY ALLOCATION IN MATCHING ALGORITHMS

Alexander Kearney

Greedy Allocations

This report aims to explore greedy allocation in the context of team-project matching algorithms and provide considerations as to reduce the negative effects greedy allocations would have in a matching system. A greedy algorithm is an optimization algorithm that, using predefined criterion, aims to make locally optimal choices based off the information available at the current moment with little to no regard for future decisions and the globally optimal solution (Harendra Kumar, 2024). Below is a graph that illustrates how a greedy algorithm would find a globally suboptimal solution through finding locally optimal solutions:



Source: Alabi (2023), Greedy algorithms: A complete guide

If the goal was to reach the end of the graph from the beginning whilst maximizing the score obtained, the optimal global solution would be following the red path to get a score of 29. A greedy algorithm however, when faced between choosing between the nodes 3 and 4, would choose the 4 as even though this will produce a less optimal global solution, it only considers the local problem of choosing the greater of 3 or 4.

If the accuracy of globally optimal solution is not required to be completely maximized, in a greedy algorithm, there is no need to work with large solution spaces as at each step it will only consider the information currently available to it, thus the algorithm can be extremely time efficient and simple to implement. Furthermore, even though greedy algorithms may not be able to find the globally optimal solution, they may be able to serve as a very quick approximate of the solution, being especially useful in situations where finding the globally optimal solution is difficult (Fiveable, 2024).

The Integer Linear Programming algorithm that was developed for the system outlined in this project is however, not an inherently greedy algorithm implementation. After determining an objective function (maximization of team-project pairing scores), the ILP algorithm considers the global solution space to determine the solution that maximizes the aforementioned objective function. Thus, instead of optimising step-by-step utilising the information currently available, the ILP algorithm attempts to ensure that the solution is globally optimal (Bradley, Hax, & Magnanti, 1977). However, there are circumstances in which the ILP algorithm can exhibit extremely greedy behaviour. As ILP aims to maximize the total score of team-project allocations, if there are large discrepancies between the b-values of teams, the algorithm may be skewed into continuously prioritizing pairing these high-performing teams with high value projects at the start of the allocation in order to satisfy the objective function, which exhibits greediness in the algorithm, and may result in lower ranking teams (with lower b-values) receiving suboptimal allocations. Thus, in order to improve this algorithm, and other future matching algorithms, a method must be considered to minimize the effect of high b-value teams causing the algorithm to exhibit greedy behaviour.

Methods for reduction of greedy allocations

Scaling benefits through normalization

One such alteration to the allocation system in order to reduce the algorithms greediness is to introduce normalization. Normalization of the b-values can be utilised to reduce the greediness in the algorithm at the expense of optimizing the efficiency of the allocation process. Normalization scales each team's benefit scores for projects relative to their own best possible outcome. This reduces the effect of disproportionate scores, preventing stronger teams from dominating simply because their absolute benefit scores are higher. In cases where some teams possess extremely high absolute benefits for specific projects, it may lead to an allocation in which the algorithm will in all cases assign that team the project; by normalizing the benefits, an attempt is made to satisfy all teams relative to their b-values and minimizing the effect of disproportionate b-values on the allocation process.

For each team T, divide the benefit value for each project for that team by the maximum benefit it can achieve across all projects, as per the equation:

$$normb(T,P) = \frac{benefit(T,P)}{maxbenefit(T)}$$

The highest b-value a team can thus get is 1, all other b-values are scaled proportionally between 0 and 1 (Bertsimas, Farias, & Trichakis, 2011). This ensures that all allocations of teams are evaluated relative to their own ideal possible project allocation, as opposed to having their allocations influenced by teams with higher absolute benefits, in which teams with very high absolute benefits would dominate in the allocation process, ensuring they get their top choice, with other teams not getting projects they may be best suited to.

This comes down to a decision being made about equity or efficiency, is the goal of the system to ensure that the projects are assigned in terms of maximizing the best fit of the overarching student body or is the goal of the system to distribute opportunities fairly between all teams? If equity is the primary focus of the system, teams with weaker absolute scores are provided with more opportunities as their normalized b-value may be higher than teams with higher absolute b-values, this however runs the risk of likely reducing the effectiveness of projects being completed by teams that may not be completed suited to them. If efficiency is chosen as the primary focus, the strongest teams are likely to dominate the allocation process for their desired project, thus leaving little to no chance for weaker teams to gain the projects they want, however it is much more likely for teams to be assigned projects in which they are more capable of completing to a satisfactory standard.

A strong team with a very high-absolute benefit that is exceptionally suited to project A and has a very high absolute benefit, and a team that is really weak across all projects, but its most suitable project is project A, would both have a normalized b-value of 1 to this project, indicating that their ability towards completing this project is equal, downplaying the first teams ability. A weak team could have high benefit scores of 1, 0.9, 0.8 towards projects when in reality they are not good fits for any of them, i.e. they are the best projects relative to the teams maximum potential, but the team is not the best for the project.

Purely ranking based on relative performance could lead to situations in which teams objectively not suited towards a project due to their very low absolute benefit scores are allocated the project even though they will likely be unable to complete it to a satisfactory standard. Thus, a possible consideration to counteract this drawback would be to implement a threshold to make sure that only teams that have achieved a minimum level of absolute benefit are considered for certain high-value, difficult projects. A project-based threshold is possible, but more a more likely option is tiered thresholds, in which projects and teams are both categorized into tiers of increasing complexity and importance for projects and absolute ability for teams. Thus, only the strongest teams compete against each other for the top projects, ensuring that the weakest teams are unable to get projects they are unsuited for, however the relative rankings from the normalization would allow for weaker teams within their own tier to compete with stronger teams within their own tier.

A potential consideration for an implementation is the use of weighted normalization as per the formula:

$$weightednormb(T,P) = a * \frac{benefit(T,P)}{maxbenefit(T)} + (1-a) * benefit(T,P)$$

This represents the balance between fairness (relative benefit) and maximizing the total benefit (absolute benefit), with a being a scaling factor that can be modified as per whether absolute or relative benefit should be prioritized. By introducing a term to factor in the absolute benefit, teams with stronger absolute benefit values are now recognized for their strength, whilst teams with weaker absolute benefit still can be recognized through their relative benefit value.

<u>Upper Confidence Bound (UCB)</u>

An alternative approach to weighted normalization is implementing an Upper Confidence Bound. A UCB is a method utilised in order to dynamically find a middle ground between team-project allocations with a very high absolute b value (labelled as exploitation in the context of UCB) whilst additionally attempting to give alternative, lower ranking teams higher ranking projects (labelled as exploration in the context of UCB and defined by the number of times the team has been considered optimal for a project and been assigned such project).

UCB is mathematically defined as:

$$UCB_{ij} = b(T_i, P_j) + \sqrt{\frac{2\ln(T)}{n_{ij}}}$$

With $b(T_i, P_j)$ being the b value between team T_i and P_j, n_{ij} being the number of times T_i has been allocated to P_j, $\sqrt{\frac{2\ln(T)}{n_{ij}}}$ being the UCB exploration term that will allow the algorithm to consider teams that have not received many optimal allocations (Princeton University, 2016). The algorithm will then iterate throughout all projects and compute a UCB score for each team using the above equation. This process will dynamically consider both fairness and optimality, as high value teams (high b-values) will be assigned projects initially due to their high $b(T_i, P_j)$ term, but if they start taking up all of the high value allocations, the $\sqrt{\frac{2\ln(T)}{n_{ij}}}$ term may increase in other teams enough that they too receive a stronger allocation, and then when such team is finally assigned a project, the n_{ij} term for their new team-project allocation is incremented for further assignments. By incorporating the exploration term, even though the highest ranking teams are initially assigned to the high value projects as in a greedy system, the exploration term defined by the UCB will cause negate this greediness by reducing the impact the high b-values have on the allocations as the algorithm now accounts for teams that have not been allocated.

Minimizing Total Score through minimization of variance

As opposed to maximizing the total score of the allocation process, which is a method that tends to prioritise the pure optimality of assigning the highest rated teams to the highest rated projects, an alternative approach would be to minimize the total score of the allocation process. If equity is considered more important than the pure optimality of the allocations, a method could be implemented that prevents teams with the most optimal absolute benefit values from being automatically allocated to the highest value projects by seeking to minimizing the score as opposed to maximizing it. Although equity and fairness would be achieved to a much higher degree for lower ranking teams, it may result in questionable fairness to the highest-ranking teams that are being matched similarly to low-ranking teams.

Such a method of minimizing the total score of allocations is to iteratively minimize the variance of allocation scores. This method will rank teams initially based on their absolute benefit values i.e. their absolute suitability to the project. The algorithm will then take these allocations, iterate through them all and then minimize the largest variation of score (the strongest team and weakest team). The algorithm will then repeat this process for the new strongest and weakest team, repeating until the variance in the allocations cannot be reduced, resulting in a set of allocations with total score being minimized.

Baseline Fairness Allocation (BFA)

BFA, or a proportional share constraint, defines a baseline share of the total benefit score available across all projects, to ensure that all teams receive their fair share. It can be defined mathematically by the equation:

$$BFA(T) = \frac{1}{n} \times \text{Total Benefit}$$

Where T is an individual team, and n is the number of teams.

All teams will be assigned at least a minimum proportion of the total benefit score, regardless of their fit scores or b values, ensuring that no team is disproportionately disadvantaged in their allocation, and that the allocation of projects is not solely based on how high a team's absolute fit for the project is, as if every high value project is given to all the high value teams immediately, the lower teams would not reach the predefined baseline of BFA.

Bottleneck Minimization

A bottleneck in this context is the lowest b-value of any team to project allocation i.e. the absolute worst team-project pairing. By focusing the algorithm on minimizing such a bottleneck, the algorithm attempts to ensure that no team is allocated a project that is extremely unsuitable. Instead of minimizing or maximizing the total score of all projects, the algorithm would ascertain the team-project pairing with the absolute lowest score, then proceed to reallocate teams to projects, trying to keep the other allocations as similar as possible, and in an attempt to increase the fairness of the allocations, would continue this process until a predetermined acceptable bottleneck value is reached, or potentially a threshold is determined in which the algorithm iterates until no team-project pairing has a score below the threshold.

This method could be paired with Tabu search, in which iterations are made to the allocations to make small variances of the allocations, in this case by changing slightly changing allocations, and storing these to memory in order to keep track of all the changes that have been made (Glover, Laguna, & Martí, 2000). This method ensures that even if a suitable solution is found, the algorithm will keep exploring other possibilities without retrying those already added to the list. By avoiding allocations that have already been made, the effects of local optima skewing the algorithm into behaving greedily can be avoided.

Conclusion

It was recognized that high variance in the b-values of teams results in greediness in allocation systems, causing an ultimately sub-optimal global solution. Possible alterations to the allocation system in order to reduce such effect of high b-values were explored such as normalization, UCB, minimizing the total score, BFA and bottleneck minimization, with all methods attempting to provide a solution to high-b values overwhelming the allocation process whilst simultaneously trying to avoid the allocations becoming suboptimal. All methods reduce the impact greedy allocation has on the system, however they come with trade-offs. Normalization ensures outlier b-values do not affect the allocations; however, it may result in weaker performing teams being assigned to high-value projects, UCB dynamically adjusts between giving allocations to high and low performance teams, however it may allocate the weakest of teams to projects they are not capable of completing. Minimization of total score and BFA will result in a system that has extremely minimal variance, meaning strong and weak teams are spread out, likely resulting in sub-optimal allocations, whereas bottleneck minimization will prioritize the overall reduction of suboptimal allocations. The choice of implementation thus depends on the individual system and what is required and prioritized in said system. A point of future research could be to explore the combination of these methods in a single system, potentially resulting in a dynamic system that best balances the strong and weak team's allocations accurately.

References

Alabi, T. H. (2023). *Greedy algorithms: A complete guide*. FreeCodeCamp. Retrieved from https://www.freecodecamp.org/news/greedy-algorithms/

Bertsimas, Farias, & Trichakis, 2011 - "The Price of Fairness":

https://www.mit.edu/~dbertsim/papers/Fairness/The%20Price%20of%20Fairness.pdf

Bradley, S. P., Hax, A. C., & Magnanti, T. L. (1977). *Applied mathematical programming: Chapter 9 - Integer programming*. Addison-Wesley. Retrieved from https://web.mit.edu/15.053/www/AMP-Chapter-09.pdf

Fiveable. (2024). *Greedy Approximation Algorithms*. Fiveable. Retrieved from https://library.fiveable.me/combinatorial-optimization/unit-8/greedy-approximation-algorithms/study-guide/LGB3wBj4i0e6Q8Xt

Glover, F., Laguna, M., & Martí, R. (2007). *Principles of Tabu Search*. Retrieved from Leeds Faculty, University of Colorado website: https://leeds-faculty.colorado.edu/glover/fred%20pubs/376%20-%20TS%20-%20Principles%20of%20Tabu%20Search%20-%20Glover,%20Laguna%20and%20Marti.pdf

Harendra Kumar. (2024). *Greedy Algorithms*. GeeksforGeeks. Retrieved from https://www.geeksforgeeks.org/greedy-algorithms/

Princeton University. (2016). *Lecture 22: Upper Confidence Bound*. COS 402: Artificial Intelligence. Retrieved from https://www.cs.princeton.edu/courses/archive/fall16/cos402/lectures/402-lec22.pdf