Chapter 1

Background

1.1 Mechanism Design

Traditionally, economists and game theorists most deeply engage with the set of problems involving the construction of systems that "fairly" (under various definitions of the word) address scenarios involving a host of participants with adverse or overlapping goals. Some examples of problems that fall under this classification are as follows: designing a voting system that successfully eliminates strategizing by any party [1], auctioning search engine advertisement spots in a way that promotes honest bidding [2], assigning roommates from a pool of students such that each student ends up with a roommate they prefer [3], and maximizing aid while allocating food donations to food banks [4].

It turns out that many mechanism design problems can be solved algorithmically, and as such computer scientists have increasingly joined the economists in the study of mechanism design theory. There are two subcategories of mechanism design relevant to this paper: stable matching and fair division.

1.1.1 Stable Matching

Matching problems are highly applicable to many real-world scenarios, such as assigning medical students to hospitals for residency [5] or the aforementioned roommates example. The canonical (and heteronormative) matching problem is known as the Stable Marriage Problem [6], in which the goal is to match an equal number of men and women such that no participant remains uncoupled. This is known as a bipartite matching problem, or one in which the participants are divided into disjoint sides and matched from one side to the other. The other important aspect of the Stable Marriage Problem is that the matching should prevent "cheating," which occurs when a man and a woman who weren't matched to each other prefer to be together over their assigned partners and will "elope," leaving the other two participants alone. This couple would be known as a blocking pair, because their elopement "blocks" the matching from being complete. Herein lies the stability aspect of stable matching: we wish to find an algorithm that outputs a stable matching of couples, with no blocking pairs.

In 1962, David Gale and Lloyd Shapley published a seminal paper for the field of stable matching in which they proposed the deferred acceptance (DA) algorithm and proved it to output a stable matching [7]. Since then, the algorithm has been adapted to fit many more scenarios with various types of considerations, including the residency problem. However, there remain subsets of problems that do not fall under the same paradigm as the DA algorithm: for instance, problems with a one-sided market. The stable roommate problem is an example of one such problem, in which participants are to be matched from the same pool rather than two distinct categories of participant and stability retains the same meaning as in the bipartite setting. Over 20 years after Gale and Shapley's paper, in 1985 Robert Irving proved stability and efficiency for an algorithm that solves the roommate problem [3]. Irving's algorithm also now serves as a jumping-off point for the majority of further research on one-sided matching problems.

1.1.2 Fair Division

The difference between matching and fair division often lies in the primary metric used to evaluate the effectiveness of the solution. Matching is most predominately studied under stability, and fair division under analysis of envy. When dividing a chocolate-vanilla cake, a person who prefers vanilla but receives chocolate will be envious of those who receive a vanilla slice. In a fair division problem, the goal is to minimize the amount of envy. The cake example is the canon example for divisible goods, but there is also a lot of study into the division of indivisible goods, which introduces a new set of considerations for maximizing welfare (the benefit participants yield from the goods they receive) and minimizing envy. The food bank example is one in which the goal pertains to fairly allocating indivisible goods [4].

Formally, envy fair division will often be evaluated under pareto-optimality. An allocation is considered pareto-optimal if there is no other allocation that is strictly better for at least one participant while retaining the same level of welfare for the other participants [8]. As more factors are added to these problems— such as considering the order in which goods arrive for distribution, scaling the number of participants, and more specific definitions of welfare—envy freeness and pareto-optimality become more difficult to ascertain.

1.2 Adding constraints

What's known as the "vanilla" version of these matching and division environments, which is to say the more simple or basic versions, have been heavily studied over years of computer science and economics literature. So, the problems are often made more difficult or made to more accurately reflect real-world problems by adding *constraints*, therefore introducing new facets of the classic problem to solve. Constraints can come in many forms, and can often make problems too difficult to solve efficiently or impossible to solve at all. In the scope of class assignment and related literature, constraints often come in the forms of quotas on class capacity and categories of

students.

1.2.1 Quotas and Diversity

Quotas in allocation and matching problems can be implemented as upper and lower bounds. Upper bound quotas implies a capacity constraint, such as a maximum number of open positions per hospital in the residency matching problem. The College Admissions problem, in which a college with n applicants can admit up to q students, is traditionally studied with these capacity constraints (where q is for quota). In fact, the College Admissions problem was introduced with Gale and Shapley's Stable Marriage DA algorithm [7]. In recent years, these quotas have been extended in a variety of ways, such as introducing lower bounds in which schools also need to accept at least m students of some type [9]. Because adding quotas can make the problem too difficult, quotas are often studied under "soft" constraints as well: for example allowing a quota to be a target range rather than a specific number [10].

Another extension of the College Admissions problem that more accurately reflects reality comes in the form of diversity constraints. In the actual college admissions problem, these constraints are often known as Affirmative Action, in which there are majority upper-bounded quotas to give space for minority students, and there are many papers that mathematically study that particular implementation as well as prove that it does not always benefit the minority applicants [11]. As a result, diversity constraints can take many forms—they have been studied under frameworks of lower-bound quotas [9], proportionality (evenly distributing types of students in the matching) [12], and group fairness (ensuring each distinct group of students is as happy with their matching as possible based on the happiness of each individual student in the group) [13].

1.2.2 Big-O

Considering constraints such as quotas and diversity often make the problem much more difficult, and in fact in many cases it becomes unfeasible to solve. Here we define "unfeasible" to mean inefficient, which we can mathematically formalize with what's known as *Big-O* notation.

Big-O notation is a way of analyzing the efficiency of algorithms in computer science. The analysis takes place *asymptotically*, or as the size of the input becomes infinitely large. It is important to analyze algorithms asymptotically so we can guarantee that an algorithm fully solves a problem efficiently, as it is less helpful if it only works on small inputs. For instance, if I as an individual have an algorithm to bake cookies (a recipe) and the steps of rolling out the dough and measuring ingredients by hand work well for a batch of 24 cookies, that is sufficient for my use case. However, if I start a cookie franchise and suddenly I need 2,400 cookies a day, my original algorithm is not going to be the most efficient way to make that many cookies, and therefore it is not a guaranteed algorithm for making cookies. It is more helpful in computer science to know our solutions will be helpful no matter the use case, instead of making many specific different algorithms for different cases.

Analyzing algorithms asymptotically allows us to estimate the running time of an algorithm, or how long an algorithm will take to terminate and output the result. In Big-O notation, we write $f(n) = O(g(n))$, where some algorithm, $f(n)$, will take at most $g(n)$ time to run. The O signifies the "at most" in that statement, or more formally it denotes an upper bound on the asymptotic running time.

Running time can be classified into different categories. The most relevant categories to this paper are constant time, polynomial time, and exponential time. Constant time, $O(1)$, can be abstractly thought of as any operation that can be completed instantly. Constant time is the fastest running time an algorithm can have. Most algorithms need to do more complicated operations than those that can be done in constant time, however, so our next category of time analysis is polynomial time. If I have cookie dough and I need to bake the cookies, my algorithm might be for every tablespoon of cookie dough roll the dough into a ball and then sprinkle cinnamon on top. That is 2 actions to complete for every tablespoon of cookie dough. So, if we have n tablespoons, we could say this step takes $O(n^2)$ time (note this would be more specifically quadratic time, but quadratic time is a subset of polynomial time). For our purposes, we can consider polynomial time to be sufficiently efficient. Our last category is exponential time, which is very slow in the world of running time and defines what we mean by "unfeasible". We consider an algorithm inefficient if it takes exponential time, such as $O(2^n)$. As n grows larger in this case, the running time increases rapidly. If we let $n = 1000$, $n^2 = 1,000,000$ which is indeed large, but not compared to 2^n, which becomes "a number much larger than the number of atoms in the universe" [14].

1.2.3 P vs NP

1.2.4 Optimization

1.3 Related Works

1.3.1 Two-sided matching

1.3.2 One-sided matching

Bibliography

[1] William S. Zwicker. "Introduction to the Theory of Voting". In: *Handbook of Computational Social Choice*. Ed. by Felix Brandt et al. Cambridge University Press, 2016, pp. 23–56.

[2] William Vickrey. "COUNTERSPECULATION, AUCTIONS, AND COMPETITIVE SEALED TENDERS". In: *The Journal of Finance* 16.1 (1961), pp. 8–37. DOI: https://doi.org/10.1111/j.1540-6261.1961.tb02789.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.1961.tb02789.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1540-6261.1961.tb02789.x.

[3] Robert W. Irving. "An efficient algorithm for the "stable roommates" problem". In: *Journal of Algorithms* 6.4 (1985), pp. 577–595. ISSN: 0196-6774. DOI: https://doi.org/10.1016/0196-6774(85)90033-1. URL: https://www.sciencedirect.com/science/article/pii/0196677485900331.

[4] Martin Aleksandrov et al. "Online Fair Division: analysing a Food Bank problem". In: *CoRR* abs/1502.07571 (2015). arXiv: 1502.07571. URL: http://arxiv.org/abs/1502.07571.

[5] Alvin E. Roth. "The Origins, History, and Design of the Resident Match". In: *JAMA* 289.7 (Feb. 2003), pp. 909–912. ISSN: 0098-7484. DOI: 10.1001/jama.289.7.909. eprint: https://jamanetwork.com/journals/jama/articlepdf/195998/jrf20024_909_912_1693348147.98749.pdf. URL: https://doi.org/10.1001/jama.289.7.909.

[6] Jon Kleinberg and Éva Tardos. *Algorithm Design: 1. Stable Matching*. Lecture Notes. 2005. URL: https://www.cs.princeton.edu/~wayne/kleinberg-tardos/pdf/01StableMatching.pdf.

[7] David Gale and Lloyd S. Shapley. "College admissions and the stability of marriage." In: *American Mathematical Monthly* 69 (1962), pp. 9–15. URL: https://www.jstor.org/stable/2312726.

[8] Sean Ingham. *Pareto-optimality*. URL: https://www.britannica.com/money/Pareto-optimality.

[9] Péter Biró et al. "The College Admissions problem with lower and common quotas". In: *Theoretical Computer Science* 411.34 (2010), pp. 3136–3153. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2010.05.005. URL: https://www.sciencedirect.com/science/article/pii/S0304397510002860.

[10] Santhini K A, Raghu Ravi, and Meghana Nasre. "Matchings under one-sided preferences with soft quotas". In: (Mar. 2025). DOI: 10.2139/ssrn.5163631.

[11] Isa E. Hafalir, M. Bumin Yenmez, and Muhammed A. Yildirim. "Effective affirmative action in school choice". In: *Theoretical Economics* 8.2 (2013), pp. 325–363. DOI: https://doi.org/10.3982/TE1135. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.3982/TE1135. URL: https://onlinelibrary.wiley.com/doi/abs/10.3982/TE1135.

[12] Thành Nguyen and Rakesh Vohra. "Stable Matching with Proportionality Constraints". In: *Operations Research* 67.6 (2019), pp. 1503–1519. DOI: 10.1287/opre.2019.1909. eprint: https://doi.org/10.1287/opre.2019.1909. URL: https://doi.org/10.1287/opre.2019.1909.

[13] Atasi Panda et al. "Group Fair Matchings using Convex Cost Functions". In: (2025). arXiv: 2508.12549 [cs.GT]. URL: https://arxiv.org/abs/2508.12549.

[14] Michael Sipser. *Introduction to the Theory of Computation*. 2nd ed. Course Technology, 2006.