3

Games People Play (With Algorithms)

THE DATING GAME

In 2013, a journalist named Amanda Lewis wrote an insightful article for *LA Weekly* about her experiences with a recently launched online dating app named Coffee Meets Bagel. One of the app's novelties was to apply the notion of economic scarcity to romantic matchmaking. Instead of encouraging users to indiscriminately spam potential dates with a barrage of online flirts, nudges, and winks, Coffee Meets Bagel limited users to a single, algorithmically proposed match or date each day, which they could accept or reject. Presumably the idea was to raise the value or demand for matches by artificially restricting the supply.

But Lewis went on to detail other "economic" side effects of the app that were perhaps less intentional, and less desirable—side effects that can be understood via game theory, the branch of economics that deals with strategic interactions between groups of self-interested individuals. Coffee Meets Bagel invited users to specify racial, religious, and other preferences in their matches, which the algorithm would then try to obey in selecting their daily proposals.

Lewis described how after not specifying any racial preferences (or, more precisely, indicating that she was willing to be matched with people from any of the site's listed racial groups), she began to receive daily matches exclusively with Asian men. The problem was that if there were even a slight imbalance in the number of women who accept matches with Asian men, and the number of Asian men, there would be an oversupply of Asian men in the app's user population. And since the matching algorithm obeys users' stated preferences, the necessary consequence was that women who did not explicitly *exclude* Asian men from their preferences would be matched with them frequently.

Given the preferences selected by the rest of the user population, Lewis's "best response" (a game theory term)—that is, her only choice if she wanted to be matched with men from other races too—was to modify her stated preferences to say that she was *unwilling* to be matched with Asian men. She reluctantly did so, even though this was not what she originally wanted. Of course, this only exacerbates the original oversupply problem, creating a feedback loop that encourages other users to do the same.

It seemed as though Lewis had been cornered into choosing between two undesirable alternatives—cornered by the stated preferences of other users, and by an algorithm that blindly and myopically obeyed those preferences for each user individually, without regard for the macroscopic consequences. At least from Lewis's perspective, the system was trapped in what a game theorist might call a "bad equilibrium." If all of the users of the app could have simultaneously coordinated to change their preferences, they might all have been happier with their resulting set of matches—but each of them individually was helpless to escape this bad outcome. It's a bit like a run on the banks in a financial crisis—even though it makes us all collectively worse off, it's still in your selfish interest to withdraw your money before it's too late.

WHEN PEOPLE ARE THE PROBLEM

There are some important similarities and differences between the dilemma Lewis found herself in on Coffee Meets Bagel and the problems of fairness and privacy considered in earlier chapters. In all three settings, algorithms play a central role—algorithms acting on, and often building predictive models from, people's data. But in algorithmic violations of fairness and privacy, it seemed reasonable to view the algorithm as the "perpetrator" and people as the "victims," at least to a first approximation. As we saw, machine learning algorithms optimizing solely for predictive accuracy may discriminate against racial or gender groups, while algorithms computing aggregate statistics or models from behavioral or medical data may leak compromising information about specific individuals. But the people themselves were not conspirators in these violations of social norms—indeed, they may not even be aware that their data is contributing to a credit scoring or disease prediction model, and may not interact with those models at all. And since the problems we identified were largely algorithmic, we could propose algorithmic solutions that were better behaved.

The Coffee Meets Bagel conundrum is more nuanced. We might argue that Lewis is also a victim of sorts—she recounts feelings of guilt when the algorithm forces her to declare what feel like racist preferences, just in order to avoid being always matched with a homogeneous group. It seems unfair in a way not dissimilar to algorithmic discrimination. But the key difference is that we can no longer place the blame exclusively or even largely on the algorithm alone—the other users, and their competing preferences, are complicit in Lewis's dilemma. After all, it wasn't the algorithm's fault that there were too many Asian men in the system relative to the population of women who reported a willingness to date them. The algorithm was simply trying to act as a mediator of sorts, attempting to satisfy each user's dating preferences in light of those of other users. We might even say that the algorithm was doing the most obvious and natural thing with the data it was given, and that the real problem was the data—the preferences themselves.

We'll eventually see that despite the complicity of users, we shouldn't let algorithms off the hook quite so easily, and that in many settings in which user preferences are centrally involved, there are still algorithmic techniques that can avoid the bad equilibrium in which Lewis became trapped. In particular, sometimes there might be multiple equilibria, and an algorithm might be able to choose, or nudge its users toward, a better one. In the case of Coffee Meets Bagel, maybe *everyone*'s preferences were like Amanda's—wanting only a diversity of matches—and everyone felt trapped into entering preferences that weren't quite truthful. Maybe a different algorithm could have done better and incentivized everyone to enter their real preferences. And in other settings we might prefer an algorithm that doesn't encourage or implement any equilibrium at all, but instead finds a solution that makes the overall "social welfare" higher. But unlike the fairness and privacy chapters, to discuss these algorithmic alternatives, we need to put the users, and their preferences, on center stage. And this in turn leads us to the powerful concepts and tools of game theory.

JUMP BALLS AND BOMBS

Many readers may have encountered a little game theory, owing in part to its generality and its ability to sometimes generate counterintuitive insights about everyday scenarios. Informally speaking, an equilibrium in game theory can be described as a situation in which all participants are acting in their own self-interest, given what everyone else is doing. The key aspect of the definition—which we'll make a bit more precise when it's called for—is the notion of selfish, unilateral stability it embodies. It is assumed that each "player" in the system (such as a user of Coffee Meets Bagel) will behave selfishly (for example, by setting or changing her dating preferences) to advance her own goals, in response to similarly selfish behavior by others, and without regard to the consequences for other players or the global outcome.

An equilibrium is thus a kind of selfish standoff, in which all players are optimizing their own situation simultaneously, and no one can improve their situation by themselves. Technically speaking, the underlying mathematical notion of equilibrium we refer to here is known as a Nash equilibrium, named for the Nobel Prize–winning mathematician and economist John Forbes Nash, who proved that such equilibria always exist under very general conditions. We'll shortly have reason to consider non-equilibrium solutions to game-theoretic interactions, as well as alternative notions of equilibrium that are more cooperative.

When equilibrium is described as a selfish standoff, it's not particularly surprising that sometimes equilibrium can be undesirable to any particular individual in the system (like Amanda Lewis), or even to the entire population. In the words of the late economist Thomas Schelling (another Nobel Prize winner), who applied equilibrium analysis to things as varied as housing choices, traffic jams, sending holiday greeting cards, and choosing a seat in an auditorium, "The body of a hanged man is in equilibrium, but nobody is going to insist the man is all right."

While the competitive, selfish nature of our equilibrium notion might seem a bit cynical or depressing—everyone is simply out for themselves, and optimizing their choices and behavior in light of everyone else's greedy behavior—it can also provide valuable clues to why and how things can sometimes go wrong in settings in which there are conflicting preferences (like racial preferences in a dating app). And it does not preclude cooperative behavior if there just so happens to be a solution in which cooperation is in everyone's self-interest. Closest to our own selfish interests, it turns out that sometimes game theory can not only describe what might go wrong at equilibrium but also can provide algorithmic prescriptions for making the outcome better.

For much of its long and storied history—depending on how one counts, the field dates to at least the 1930s—game theory trafficked primarily in the precise understanding of simple and highly stylized versions of real-world problems. These stylizations could often be described by small tables of numbers specifying the payoffs of just two players (and therefore their preferences, since it is assumed that a player will always prefer whatever offers their highest payoff, in light of the opponent's behavior). Classic examples include Rock-Paper-Scissors (useful in the real world as an alternative to jump balls in recreational basketball), where, for instance, choosing Rock yields payoff +1 against Scissors, which in turn receives payoff -1. The equilibrium turns out to be both players uniformly randomizing among their choices, playing Rock, Paper, and Scissors with probability 1/3 each. This is the only solution with the aforementioned unilateral stability property—if I uniformly randomize, your best response is to do so as well, and if you do anything else (such as playing Paper even slightly more often than the other two choices), I'll exploit that and punish you (by always playing Scissors). Some readers may be even more familiar with Prisoner's Dilemma, another simple game that has a disturbing equilibrium in which both players sabotage each other to their mutual harm, even though there is a cooperative non-equilibrium outcome in which they both benefit. As the story goes, two accomplices to a crime are captured and held in separate cells. They can either "cooperate" with their accomplice and admit to nothing or "defect" and admit to the crime and testify against their partner. If your partner defects and testifies against you, you get a long sentence. If your partner cooperates, you get only a short one. And if you defect, the prosecutor offers to shave a little bit off the sentence you would have otherwise received. The problem is that if I cooperate, you can do even better by sabotaging me and defecting, and vice versa. When we both defect, we each experience close to the worst possible outcome. But since mutual cooperation is not unilaterally stable, we drag each other into the sabotaging abyss of equilibrium, hence the "dilemma".

Despite the simplicity of such games, they have occasionally been applied to rather serious and high-stakes problems. During the Cold War, researchers at the RAND Corporation (a long-standing think tank for political and strategic consulting) and elsewhere used game-theoretic models to try to understand the possible outcomes of US-Soviet nuclear war and détente—efforts that were memorably if darkly lampooned in the 1964 Stanley Kubrick film *Dr. Strangelove,* which ends with the Prisoner's Dilemma–like nuclear annihilation of the world. But the lasting influence and scope of game theory (which has also been widely applied to evolutionary biology and many other fields far from its origins) bears testament to the value of deeply understanding a "toy" version of a complex problem. By distilling strategic tensions down to tables of numbers with maybe only a few rows and columns, game theorists could solve exactly for the equilibrium and try to understand its ramifications for the real problem—which was usually considerably more complicated, messy, and imprecise.

As we shall see, the technological revolution of the last two decades has considerably expanded the scope and applicability of game-theoretic reasoning, while at the same time challenging the field to tackle problems of unprecedented scale and complexity—problems involving sophisticated algorithms operating on rich datasets generated by thousands, millions, or sometimes billions of users. Reducing such problems to simple models of the Rock-Paper-Scissors or Prisoner's Dilemma variety is entirely infeasible and would throw away too much salient detail to be even remotely useful. The matchmaking equilibrium determined by the dating preferences of the users of Coffee Meets Bagel simply isn't something that can be computed by hand and understood with just a few numbers. It would itself require an algorithm to compute, which in an informal sense is exactly what the app provides.

To tackle such challenges, the new field of algorithmic game theory has emerged and developed rapidly. It blends ideas and methods from classic game theory and microeconomics with modern algorithm design, computational complexity, and machine learning, with the goal of developing efficient algorithmic solutions to complex strategic interactions between very large numbers of players. At a minimum, it aspires to broadly understand what might happen in systems like Coffee Meets Bagel. At its best, it is not only descriptive but also prescriptive—as in the fairness and privacy chapters, telling us how to design socially better algorithms, but now in settings in which the incentives and preferences of users, and how we will examine in act on them, must be taken into account. These are the topics we will examine in this chapter.

THE COMMUTING GAME

To illustrate how the scale and power of modern technology have made algorithmic game theory relevant, let's consider an activity that many people engage in every day but may never have thought about as a "game" before: driving a car. Suppose you live in a busy metropolitan area with congested roads, and each day you must drive from your house in the suburbs to your workplace downtown. There is a complex network of freeways, highways, streets, thoroughfares, and back roads you must navigate, and the number of plausible routes you could take might be very large indeed. For instance, maybe the most straightforward route is to get on the freeway at the entrance nearest your home, get off at the exit nearest your workplace, and drive on the main surface streets before and after the freeway. But maybe one segment of the freeway often has bumper-to-bumper traffic during your commute time, so sometimes it's better to get off earlier, take some back roads through a residential neighborhood, and rejoin the freeway later. And on any given day, transient conditions—a traffic accident, road construction or closures, a ball game—might render your usual route much slower than some other alternative.

If you think about it, on a moderately long commute in a busy city the number of distinct routes you might take or at least try over time could be in the dozens or even hundreds. Of course, these different routes might overlap to varying degrees—maybe many of them use the freeway, and if you live on a cul-de-sac, they will always start by getting off your street—but each route is a distinct path through your local network of roads. In game theory terminology, your "strategy space"—the possible actions you might choose—is much larger than in simple games like Rock-Paper-Scissors, where by definition you only have three actions available.

So you have a lot of choices; but what makes this a "game"? It's the fact that if you're like most commuters, your goal or objective is to minimize your driving time. But the driving time on each of your many possible routes depends not just on which one you choose but also on the choices of all the other commuters. How crowded each route is determines your driving time as much or more than the length of the roads, their traffic lights, speed limits, and other fixed aspects. The more drivers who choose a given road, the longer the driving time for all routes that use that road, making them less attractive to you. Similarly, the fewer drivers there are on a road, the more you might want to choose a route that uses it (as long as the other segments on the route aren't too busy).

The combination of your hundreds of possible routes with the choices made by the tens of thousands of other commuters presents you with a well-defined, if mind-boggling, optimization problem: pick the route with the lowest total driving time, given the choices of all the other drivers. This is your "best response" in the commuting game. And it's not at all unreasonable for us to assume you will at least try to act selfishly and choose your best response (just as Amanda Lewis begrudgingly did on Coffee Meets Bagel). Who wants to spend more time commuting than they need to?

Note that while the complexity of this game is much greater than something like Rock-Paper-Scissors, the fundamental commonality is that the payoff or cost of any individual player's choice of action depends on the action choices of *all* the players. There are important differences as well. In Rock-Paper-Scissors the two players have the same payoff structure, whereas if you and I live and work in different places, our cost structures will differ (even though we still both want to minimize driving time for our own commute). And if you and I commute at different times of day, we aren't really even in the same round of the game. But these differences don't alter the fundamental view of commuting as just another (albeit very complex) game. And this means that as with Rock-Paper-Scissors and Coffee Meets Bagel, it makes sense—from both qualitative and algorithmic perspectives—to discuss its equilibrium, whether it is "good" or "bad," and whether there might be a better outcome.

YOUR SELFISH WAZE

Commuting has been the game we have described ever since roads became sufficiently congested that the choices of other drivers affected your own. But for many years this formulation wasn't particularly relevant, because people really didn't have the ability to truly or even approximately optimize their route based on the current traffic. Commuting was a game, but people couldn't play it very well. This is where technology changed everything—and, as we shall see, not necessarily for the collective good.

The first challenge in playing the commuting game is informational. As older readers will recall, for decades you had to plan your daily commute by cobbling together radio and television traffic reports that were both incomplete (perhaps covering only major freeways, and providing little or no information about the vast majority of roads) and inaccurate (since the reports were only occasional, perhaps on the half hour, and therefore often stale). But even if one could magically always have perfect and current traffic data for every road, there is a second, algorithmic challenge, which is computing the fastest route between two points in a massive network of roadways, each annotated by its current driving time.

In a relatively short period, navigation apps such as Waze and Google Maps have effectively solved these problems. The algorithmic challenge was actually the easier one—there have long been fast, scalable algorithms for computing fastest routes (or "shortest paths," as they are called in computer science) from known traffic. A classical one is Dijkstra's algorithm, named for the Dutch computer scientist who described it in the late 1950s. Such algorithms in turn allowed the informational problem to be solved by crowdsourcing. Even though early navigation apps operated on traffic data not much better than in the pre-Internet days, they could still at least suggest plausible routes through a complex and perhaps unfamiliar city—a vast improvement over the era of dense and confusing fold-up maps in the glove compartment. And once users started adopting the apps and permitting (wittingly or not) their location data to be shared, the apps now had thousands of real-time traffic sensors right there on the roadways.

This crowdsourcing was the true game-changer. Whatever pride you might have had in your navigational wizardry in your home city, the utility of a tool that automatically optimized your driving time in response to real-time, highly accurate, and granular traffic data on virtually every roadway anywhere was just too alluring to decline. User populations grew to the hundreds of millions, further improving traffic data coverage and accuracy.

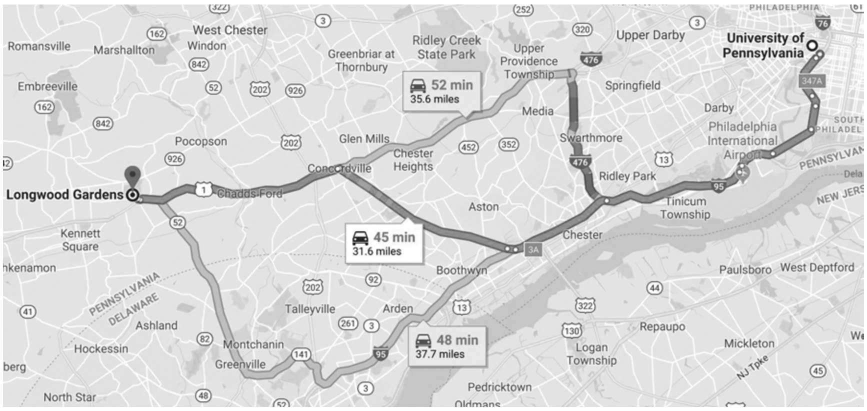


Fig. 17. Typical screenshot from Google Maps, showing just a few of the many hundreds or thousands of routes between two locations in the greater Philadelphia area. The suggested routes are ranked by lowest estimated driving time.

1 From our game-theoretic viewpoint, modern navigation apps finally allowed any player in the commuting game to compute her best response to all her "opponents" on the roads, anywhere and anytime. And there is little doubt that these apps are extraordinarily useful and efficient, and are doing the most obvious thing with the massive data at their disposal: looking out for the best interests of each individual user, finding their fastest route in light of the current traffic patterns.

THE MAXWELL SOLUTION

2 But there is another perspective worth considering, which is that because the apps are computing best responses for every player individually, they are driving the collective behavior toward the kind of competitive equilibrium we have discussed in Coffee Meets Bagel, Prisoner's Dilemma, and Rock-Paper-Scissors—the apps enable, and thus encourage, selfish behavior on everyone's part. And with Coffee Meets Bagel and Prisoner's Dilemma, we already have seen cases in which the resulting competitive equilibrium may not be something that any particular individual is happy about. Surely anyone with even moderate experience with city driving has encountered situations in

which individually selfish behavior by everyone seems to make everyone worse off—for instance, in the jockeying and slithering that occur when merging down to a few lanes of traffic at the entrance to the Lincoln Tunnel in New York City.

What might be an alternative to individually selfish, collectively competitive driving? Surely no one believes we'd all be better off (at least in terms of driving time) if we rolled back the calendar and returned to the era of spotty traffic reports and folding maps. But now that we do have large-scale systems and apps with the ability to aggregate granular traffic data, compute and suggest routes to drivers, it might be worth considering making recommendations other than the obvious, selfish ones.

Let's consider a conceptually simple thought experiment. Imagine a new navigation app—we'll name it Maxwell, for reasons that will become clear later—that behaves similarly to Google Maps and Waze, at least at a high level. Like those apps, Maxwell gathers GPS and other location data from its users to create a detailed and up-to-date traffic map, and then for any user at any moment, it computes and suggests a driving route based on origin, destination, and the traffic. But Maxwell is going to use a very different algorithm to compute suggested routes—an algorithm with a different goal, and one that will lead to a different and better collective outcome than the competitive equilibrium.

3 Instead of always suggesting the selfish or best response route to each user in isolation, Maxwell gathers the planned origin and destination of every user in the system and uses them to compute a coordinated solution that is known in game theory as the maximum social welfare solution (hence the app's name, Maxwell). In the commuting game, the maximum social welfare solution is the one that minimizes the *average* driving time across the entire population, instead of trying to minimize the driving time of each user *individually* in response to the current traffic. By minimizing average driving time, Maxwell is maximizing the time people have to do other things, which is presumably a good thing.

It might seem like there shouldn't be any difference between these two solutions, but there is. A stylized but concrete example will be helpful here. Imagine that there is a large population of *N* drivers in a city, and all of them want to simultaneously travel from location A to location B. There are only two possible routes from A to B; let's call them the slow route and the fast route.

The slow route passes many schools, hospitals, libraries, restaurants, shops, and other places that generate a great deal of pedestrian traffic. It is littered with stop signs, crosswalks, speed bumps, and police making sure that all laws are obeyed. Because of this, it really doesn't matter how many drivers take the slow route. The real bottleneck is all the stop signs, crosswalks, speed bumps, and police. In other words, we are going to assume that the time it takes to travel from A to B on the slow route is independent of the number of drivers on it. To make things concrete, let's suppose that travel time is exactly one hour.

The fast route, on the other hand, is a freeway without speed limits or police, but it has limited capacity. If you're the only one driving on it, it can be very fast—almost instantaneous—to get from A to B. But the more drivers who take the fast route, the less fast it becomes. Specifically, let's assume that if *M* out of the *N* drivers take the fast route, the travel time for each of them is M/N hours. Since *M* is a whole number less than or equal to *N*, this means that the time it takes to travel the fast route is between $1/N$ (if only one driver takes

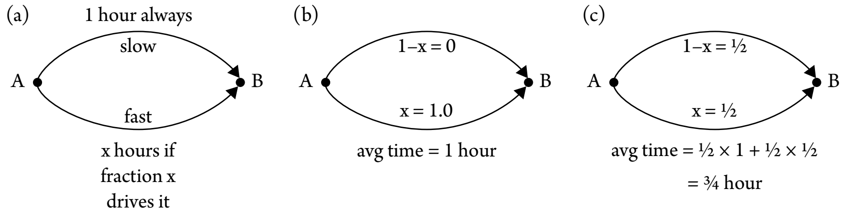


Fig. 18. Illustration of simple two-route navigation problem, with a fixed-driving-time slow route and a traffic-dependent fast route (a); equilibrium or Waze solution (b); Maxwell solution (c).

it), which is nearly zero if N is large, and N/N, which is one hour if everyone takes it. So in the worst case, the fast route isn't any faster than the slow route, but it depends on M. From your perspective as a driver, you'd like all the other $N - 1$ drivers to take the slow route, taking exactly an hour each, and for you to take the fast route and virtually teleport to the destination. Of course, none of the other drivers like your solution.

Now let's analyze the consequences of selfish behavior, of the kind enabled and even encouraged by existing navigation apps. If we think about it, such apps will recommend the fast route to the entire population of N drivers. This is because if the app recommended the slow route to even a small number of drivers—say, five—then these five drivers all experience the fixed one hour of slow route travel time, but any one of them would have been slightly better off taking the fast route, where the travel time will be $(N - 5)/N = 1{-}5/N$ hours—just a shade less than an hour. So the competitive equilibrium that results from selfish routes is where everyone takes the fast route, which then becomes no faster than the slow route, and everyone's driving time is exactly one hour. Note that in this equilibrium, each individual driver is actually indifferent to which route is taken—the driving time for both is an hour—but if even one driver is on the slow route, the drivers on the fast route are strictly better off.

What is Maxwell going to do in the same situation? It is going to pick half of the drivers—let's say a random half—and suggest that they drive the slow route, and suggest the fast route to the other half. Before discussing why anyone would follow the suggestion to drive the slow route, let's analyze the average driving time in this alternative solution. Obviously the $N/2$ drivers taking the slow route will, as always, experience a driving time of one hour. The $N/2$ drivers taking the fast route will experience a driving time of only $(N/2)/N = 1/2$ hour each. So the average driving time across the entire population is $(1/2 \times 1) + (1/2 \times 1/2) = 3/4$ of an hour, or only forty-five minutes. It turns out this is the split of the population into the slow and fast route that minimizes the average driving time. (For readers who both took and remember some calculus, if we let x denote the fraction of the population on the fast route, the average driving time is simply $1 - x + x^2$, which is minimized at $x = 1/2$ and yields the 3/4 hour average.)

In other words, by suggesting routes with a different goal—one with an explicit concern for the collective benefit rather than individual self-interest—we can reduce the overall driving time significantly, by 25 percent in this case. And we can do so without making anyone worse off than they would have been in the competitive equilibrium. So there is a better alternative to the competitive equilibrium in our toy example, and the gains may generally be even greater in complex networks of roads in the real world.[1] (In 2018 a team of researchers from UC Berkeley presented empirical evidence that navigation apps indeed cause increased congestion and delays on side streets.) The question is whether and how we can actually realize this savings of collective driving time "in the wild." 4

MAXWELL'S EQUATIONS

The first challenge in implementing Maxwell is algorithmic. While it was a simple calculus exercise to find the socially optimal solution in our toy two-route example, how will Maxwell do it when confronted with colossal networks of real roads and freeways, and thousands or more drivers, all with different origins and destinations? At least the selfish routes suggested by Google Maps and Waze can be computed quickly on large-scale networks, using Dijkstra's algorithm. 5

Fortunately, it turns out that there are also fast, practical algorithms for computing the global solution that minimizes collective average

[1] A distinct but related side effect of selfish behavior in commuting is known as Braess's paradox, in which adding capacity to a network of roadways actually *increases* congestion (or closing roads decreases congestion), and which has been reported to have occurred in large cities such as Seoul, Stuttgart, and New York City. Such phenomena cannot occur under the Maxwell solution.

driving time in large networks, especially if the driving times on each road are a linear (i.e., proportional) function of the number or fraction of drivers on them (like the roads in our earlier example, or more realistic ones such as a road that hypothetically takes $1/4 + 2x$ hours to travel if a fraction x of the population drives on it). This proportional model actually seems like a reasonable one for real traffic, and we can easily envision deriving such models from the voluminous empirical data that services such as Waze already routinely collect, which provides samples of the driving times at different levels of traffic. And for such roads, the average driving time is then just a quadratic function (e.g., if a fraction x of drivers takes a $1/4 + 2x$ road, then the contribution to the overall average driving time from just this road will be $(1/4 + 2x)x = 1/4x + 2x^2$).

Even though Maxwell must solve a very high-dimensional problem—finding the exact fractions of drivers taking every road in the network, in a way that is consistent with everyone's origins and destinations and is socially optimal—it is a problem of a well-studied and well-understood type that has very practical algorithms. It is an instance of what are known as convex minimization problems, which can be solved by so-called gradient descent methods; this is just algorithm-speak for "walk downhill in the steepest direction to quickly get to the lowest point in the valley." In our context, this simply means that we start with an arbitrary assignment of driving routes and make incremental improvements to it until the collective driving time is minimized.

What if the driving times on the roads are not proportional to traffic but are more complex functions? For example, consider a hypothetical road whose driving time is $x/2$ for $x < 0.1$ but is $10x + 2$ for $x >= 0.1$. So the time it takes to drive this road takes a sudden, discontinuous jump once 10 percent or more of the population takes it. For more complex roads such as these, we do not know of fast algorithms that are always guaranteed to find the socially optimal solution, but we do know of good techniques that work well in practice. And in these more complex cases, the improvement of the socially optimal solution over the selfish equilibrium can be much greater than in the proportional-road setting. Thus at least the algorithmic challenges in implementing Maxwell seem surmountable.

CHEATING ON MAXWELL

But as is often the case in settings in which human preferences and game theory are involved, the biggest challenge Maxwell would face in the real world has less to do with good algorithms and more to do with 6 incentives. Specifically, why would anyone ever follow the advice of an app that sometimes doesn't suggest the route that would be fastest for him at that given moment? Consider any particular driver assigned to the slow route in the earlier example—he could always "defect" to the fast route and reduce his driving time, so why wouldn't he? And if everyone did this, they all revert to the competitive equilibrium of existing apps.

If we think about it for a moment, it seems possible that even current navigation apps such as Google Maps and Waze could also be susceptible to various kinds of cheating or manipulation. For example, I could lie to Waze about my intended origin and destination, in an effort to influence the routes it recommends to other users in a way that favors me—creating false traffic that causes the Waze solution to route other drivers away from my true intended route. Manipulation of this variety apparently occurred in residential Los Angeles neighborhoods frustrated by the amount of Waze-generated traffic, as reported by the *Wall Street Journal* in 2015:

7 Some people try to beat Waze at its own game by sending misinformation about traffic jams and accidents so it will steer commuters elsewhere. Others log in and leave their devices in

their cars, hoping Waze will interpret that as a traffic standstill and suggest alternate routes.

But the incentive problems that Maxwell faces are arguably even worse, because they are not simply about drivers lying to the app; rather, the problem is drivers disregarding its recommendations entirely when they are not best responses.

There are a couple of reasonable replies to this concern. The first is that we may eventually (perhaps even soon) arrive at an era of mostly or even entirely self-driving cars, in which case the Maxwell solution could simply be implemented by centralized fiat. Public transportation systems are generally already designed and coordinated for collective, not individual, optimality. If you want to fly commercially from Ithaca, New York, to the island town of Lipari in Italy, you can't simply direct American Airlines to take a nonstop route along the great circle between the two locations—instead you'll have multiple flight legs and layovers, all for the sake of macroscopic efficiency at the expense of your own time and convenience. In a similar vein, it would be natural for a massive network of self-driving cars to be coordinated so as to implement navigation schemes that optimize for collective average driving time (and perhaps other considerations, such as fuel efficiency) rather than individual self-interest.

But even before the self-driving cars arrive en masse, we can im-
agine other ways Maxwell might be effectively deployed. One is that
if, as in our two-route example above, Maxwell randomly chooses the
8 drivers who are given nonselfish routes, users might have a stronger
incentives to use the app, since over time the assignment of nonself-
ish routes will balance out across users, and then each individual user
would enjoy lower average driving time. So while you might have an
incentive to disregard Maxwell's recommendation of a slower route
on any given trip (which you might well discover by using Google
Maps to see your selfish best-response route), you know that over

time you will benefit from following Maxwell's suggestions (as long as others do as well). We might call this phenomenon cooperation through averaging, which is also known to occur when human subjects play repeated rounds of Prisoner's Dilemma. But perhaps there is a better and more general solution to these incentive concerns.

COOPERATION THROUGH CORRELATION

Let's review where we are. Maxwell may have a better collective solution, but it is vulnerable to defection, and even the selfish navigation apps may be prone to manipulation. Both approaches have good algorithms, but the concern is that their goals can be compromised by human nature.

It turns out that sometimes these concerns can be overcome by considering yet a third notion of solution in games (our first being the selfish equilibrium, and the second being the best social welfare but non-equilibrium Maxwell solution). This third notion is known as a correlated equilibrium, and it too can be illustrated by a simple situation involving driving. Imagine an intersection of two very busy roads, one of which has a yield sign and the other of which does not. Then not only the law but also the selfish equilibrium is for drivers on the yielding road to always wait for an opening before continuing, and for drivers on the through road to speed along. Given what the drivers on the other road are doing, everyone is following their best response. But drivers on the yielding road suffer all the waiting time, which might feel unfair to them.

In this example a correlated equilibrium could be implemented with a traffic signal, which now allows drivers to follow strategies that depend on the signal, such as "If the light shows green to me, I will speed through, and if it shows red to me, I will wait." If everyone follows this strategy, they are all best-responding, but now the waiting time is split between the two roads—a fairer outcome not possible with only yield signs. The traffic signal is thus a coordination (or correlating) device that allows cooperation to become an equilibrium.

Can cooperation via coordination help solve Maxwell's incentive problems? The answer is yes—at least in principle. Very recent research has shown how it is possible to design a variant of Maxwell's algorithm—let's call it Maxwell 2.0—that quickly computes a correlated equilibrium, and enjoys three rather strong and appealing incentive properties. First, it is in the best interest of any driver to actually use Maxwell 2.0: nobody has an incentive to opt out and use another app instead (unlike Maxwell 1.0). Second, it is in the best interest of any driver to honestly input his true origin and destination: one cannot beneficially manipulate the solution found by Maxwell 2.0 by lying to it. (This is a property known as "truthfulness" in game theory.) Third and finally, it is in the best interest of any driver to actually follow the route recommended by Maxwell 2.0 after he sees it. So all drivers want to use Maxwell 2.0, and to use it honestly—both in what they input to it and in following its output.

How does Maxwell 2.0 achieve these apparently magical properties? Looking back to Chapter 1, it does so by applying differential privacy to the computation of the recommended routes in a correlated equilibrium. Recall that differential privacy promises that the data of any individual user cannot influence the resulting computation by very much. In this case a user's data consists of both the origin and destination he reports to Maxwell 2.0 and the traffic data his GPS locations contribute. The computation in question is the assignment of driving routes in a correlated equilibrium. Since a single driver's data has little influence, it means manipulations like lying about where you want to go or leaving the app on in your parked car won't benefit you or change what others do. And since a correlated equilibrium is being computed, your best response is to follow the suggested route.

Note that there was no explicit privacy goal here. Rather, the incentive properties we desired were a by-product of privacy. But at a high level, it makes sense: if others can't learn anything about what you have entered into Maxwell 2.0 or what it told you to do, then you can't beneficially manipulate your inputs to change the behavior of others. That techniques developed for one purpose (like privacy) turn out to have applications elsewhere (like incentivizing truthfulness) is a common theme in algorithms. In fact, differential privacy has many other non-privacy-related applications—we will see another one in Chapter 4.

GAMES EVERYWHERE

Even though Maxwell is only a hypothetical app (at least for now), we spent some time on the commuting game because it crisply illustrates a number of more general themes, and does so in a situation with which many of us have daily experience. These themes include:

- Individual preferences (e.g., where you want to drive from and to) that may be in conflict with those of others (e.g., traffic).
- The notion of a competitive or "selfish" equilibrium, and the observation that convenient modern technology (e.g., Waze) might drive us toward this equilibrium.
- The observation that there might be socially better outcomes that can be found with fast algorithms (e.g., Maxwell) that also enjoy good incentive properties (e.g., Maxwell 2.0).
- The lesson that when an app is mediating or coordinating the preferences of its users (as opposed to simply using their data for some other purpose, such as building a predictive model), the algorithm design must specifically take into consideration how users might react to its recommendations—including trying to manipulate, defect, or cheat.

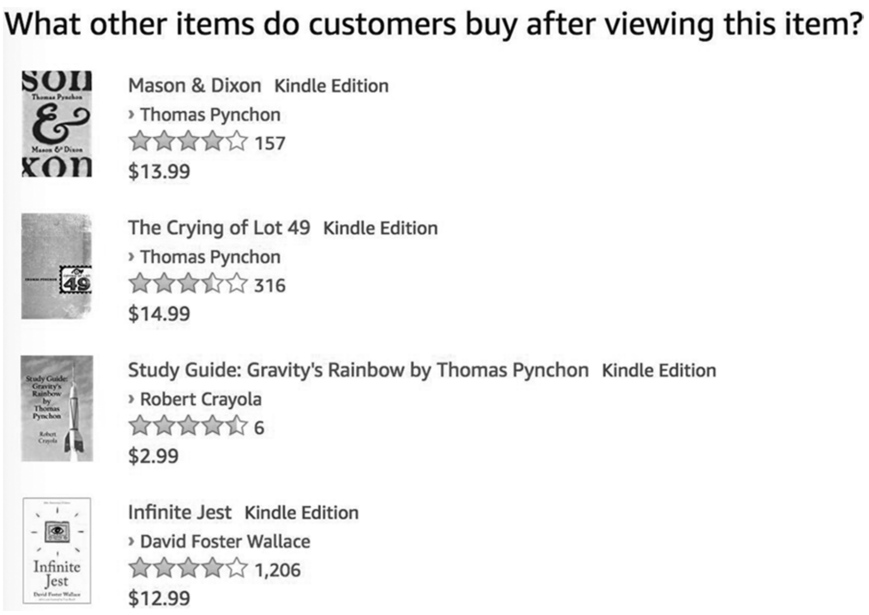
In the rest of this chapter, we'll see that these same ideas apply in a wide variety of other modern, technology-mediated interactions, from routine activities such as shopping and reading news to more specialized situations such as assigning graduating medical students to hospital residencies, and even to kidney transplants. In some cases we'll see that the algorithms involved might be pushing us toward a bad equilibrium, and in other cases we'll see they are doing social good. But in all of them, the design of the algorithm and the preferences and desires of the users are inextricably intertwined.

SHOPPING WITH 300 MILLION FRIENDS

Like driving, shopping is another activity that many of us engage in daily and that has been made more social and game-theoretic by technology. Before the consumer Internet, shopping—whether for groceries, plane tickets, or a new car—was largely a solitary activity. You went to physical, local stores, and your decisions were based on your own experience and research (and perhaps the advertising you were exposed to). For bigger purchases such as a car or a television, there might be publications such as *Consumer Reports*. But for most things, you were more or less on your own. As in the commuting game, you had shopping preferences—admittedly more complex, multifaceted, and harder to articulate than simply wanting to drive from point A to point B. But there were very few tools to help you optimize your decisions. It was the shopping equivalent of the era of spotty traffic reports and fold-up maps.

As readers will have experienced, all of this changed with the explosive growth of online shopping. Once we began researching and purchasing virtually everything imaginable on the web, we provided retailers such as Amazon extremely fine-grained data on our interests, tastes, and desires. And as we have discussed in previous chapters, machine learning could then take this data and build detailed predictive models that generalized from the products and services we already did like to the ones we would like if only we were made aware of them. The technical term in the computer science community for this general technology is *collaborative filtering* (which was widely used in the Netflix competition, whose privacy violations were discussed in Chapter 1). It's worth understanding a bit about the algorithms and models involved, and how they have become more sophisticated and powerful over time.

First of all, the "collaborative" in collaborative filtering refers to the fact that not only will your own data be used to make recommendations to you, but so will everyone else's data. The most rudimentary techniques rely just on counting, and appear in the form of Amazon's "customers who bought this item also bought" suggestions. By simply keeping statistics on items frequently purchased together or in quick succession by other users, Amazon can tell you what you might want or need with your current purchase. These kinds of counting recommendations aren't really generalizing in any deep or meaningful way—one just needs to look up the historical frequency statistics—and they tend to be relatively obvious, like tennis balls to go with your new racquet, or a David Foster Wallace novel to go with your purchase of Thomas Pynchon's *Gravity's Rainbow*.


What other items do customers buy after viewing this item?

Mason & Dixon Kindle Edition
› Thomas Pynchon
157
$13.99

The Crying of Lot 49 Kindle Edition
› Thomas Pynchon
316
$14.99

Study Guide: Gravity's Rainbow by Thomas Pynchon Kindle Edition
› Robert Crayola
6
$2.99

Infinite Jest Kindle Edition
› David Foster Wallace
1,206
$12.99

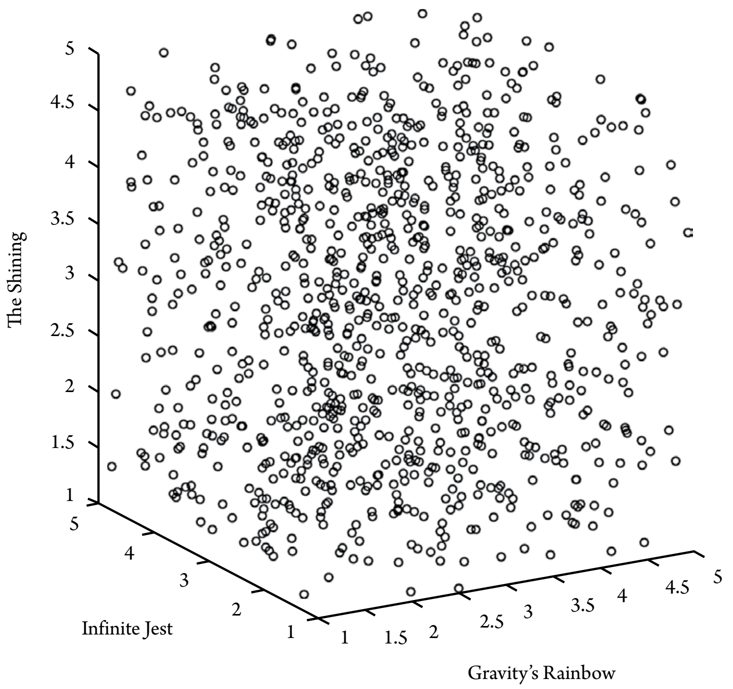
Fig. 19. Amazon recommendation for products related to the Thomas Pynchon novel *Gravity's Rainbow*, based on simple customer purchase statistics.

This level of collaborative filtering might capture similar or related products but has no explicit notion of similar *people*. Instead of just suggesting balls to go with racquets, how can we solve harder problems, like suggesting vacation destinations to someone who has only ever purchased books online? If your taste in reading reveals enough about what type of person you are, then we can suggest the vacations taken by others like you.

SHOPPING, VISUALIZED

What's a principled, algorithmic way of mapping users to their "types" based on their shopping history, and how would we discover these types in the first place? Let's imagine a greatly simplified world in which there are only three products for sale on Amazon, all of them books: Thomas Pynchon's *Gravity's Rainbow*, David Foster Wallace's *Infinite Jest*, and Stephen King's *The Shining*. Suppose there are one thousand Amazon users, each of whom has rated all three of these books on a continuous scale of 1 to 5 stars. (We'll use a continuous scale—which allows ratings like 3.729—because it will make visualization easier, but everything we'll say applies equally well to the discrete rating systems common on platforms like Amazon.) Then we can actually plot and visualize each user as a point in three-dimensional space. For instance, a user who rated *Gravity's Rainbow* as a 1.5, *Infinite Jest* as a 2.1, and *The Shining* as a 5 would have an *x* coordinate value of 1.5, a *y* value of 2.1, and a *z* value of 5. Together the one thousand users give us a cloud of points in space.

What might we expect this cloud to look like? If there were absolutely no correlations in the population ratings—that is, if knowing a user's ratings for one of the three books gives us no information or insight as to how much they might like the other two—then we would expect this cloud to look rather diffuse and spread out. For instance, if there are no correlations, and for each book the user ratings are evenly distributed between 1 and 5, the cloud would essentially fill up the three-dimensional space, as in the top panel of Figure 20.



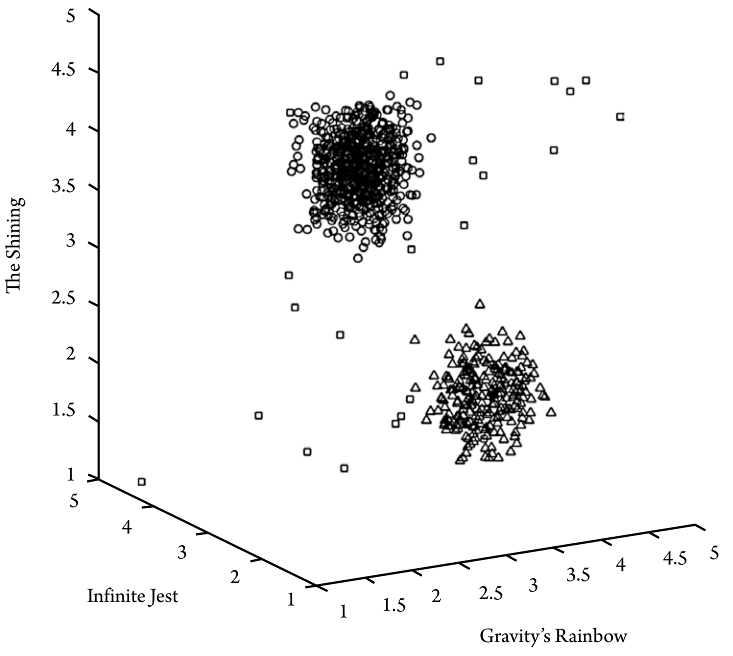


Fig. 20. Visualization of hypothetical user ratings of three products. On the left, ratings across products are uncorrelated, and each product is equally likely to be rated anywhere from 1 to 5. On the right, there are strong correlations across products, and some products have higher average ratings than others.

But we might well expect the cloud to look quite a bit less random and more structured. In particular, suppose it's the case that readers of *Gravity's Rainbow* are, on average, much more likely to enjoy *Infinite Jest* than *The Shining*; conversely, fans of *The Shining* tend to find both *Gravity's Rainbow* and *Infinite Jest* a bit obtuse and overwrought. Then the ratings would actually form two distinct clouds or clusters of points—one cloud with high ratings for *The Shining* and low ratings for the other two (the cloud of circles in the bottom panel of Figure 20), and the other with high ratings for *Gravity's Rainbow* and *Infinite Jest,* and low ratings for *The Shining* (the cloud of triangles in the bottom panel of Figure 20). The structure in this example jumps right out at us even though there are some users (the squares) that don't really fall into either of the two clouds; they are outliers that are not as easily categorized.

Perhaps unsurprisingly, the real world looks quite a bit more like the bottom panel than the top panel—as we have mentioned elsewhere, there really are very strong correlations in the products people buy or rate (as well correlations between more abstract things, such as between our political beliefs, race, religious practices, and sexual preferences). Before turning to the question of how we might identify these clouds algorithmically, let's appreciate two very powerful properties they enjoy.

First, suppose we are given a user who has only rated *Infinite Jest* and given it a 4.5. We can't map this user into our three-dimensional space because we are missing two of the ratings. But if we instead want to *recommend* a new book to this user, the data makes it intuitively clear what we should do: when we project the data onto the *Infinite Jest* axis, it is quite likely that this user is in reality a member of the triangle cloud and not the circle cloud and thus would enjoy *Gravity's Rainbow* much more than *The Shining*. So knowing about the two clouds lets us generalize, or infer what new products this user might like. (If instead we just know a user liked *The Shining,* we don't have a book to recommend, but at least we know two books *not* to recommend.)

The second nice property is that the data itself automatically tells us what the user "types" are. We don't have to guess or posit the existence of prefabricated categories such as "postmodern fiction lover" or "horror fan," or identify books as postmodern or horror. We don't even have to give names to the types at all—they are just groups of users whose tastes across products tend to be similar.

Incidentally, the attentive reader might note than in our three-book example, it's not obvious why it's helpful to identify user types or clusters, instead of just making the simplistic product correlation recommendations discussed earlier ("People who bought *Gravity's Rainbow* often bought *Infinite Jest*"). That's because the benefits of user modeling really arise when there are far more products than just three. Say you've rated or bought one hundred products on Amazon so far. Instead of picking one of your purchases and suggesting a related product, we use all one hundred of your purchases to identify your type first, and then use your type to suggest things you might like from the entire universe of products. This is how it is possible to recommend vacation destinations, rather than just books, to someone who has only ever purchased books.

A DIFFERENT KIND OF CLOUD COMPUTING

In the example above, simple visualization of the data made the two clouds or types of users jump out at us—we could just "eyeball" them. But not only will this not scale up when we have 500 million products (the approximate number sold on Amazon) instead of three, it's not even well-defined. We need a more algorithmic specification of the problem.

Let's still represent users as points in product rating space, but now that space will be much higher-dimensional—perhaps not all 500 million Amazon products, but maybe a very large sample of diverse and representative ones. Given our cloud of user rating points, one natural formulation of our problem is the following:

> Find a division of users into *K* groups such that the average distance between users in the same group is much smaller than the average distance between users in different groups.

In our simple example above, if we choose $K = 2$, it's pretty clear that the groups that optimize this criterion are the circle and triangle groups we identified; for completeness we need to assign the outlier black points to whichever cloud they are closest to, circle or triangle. But if outliers are rare, they won't influence the optimal solution very much. Furthermore, at least in this extreme example, the data also tells us what the "right" value of *K* is—if we increase to $K = 3$, we'll perhaps either put a square outlier in its own private cloud, or maybe split the circle or triangle cloud in half, but neither of these would improve our solution much, so we should stop at $K = 2$.

For a large number of users in a large space of products, the problem we have identified above can be computationally challenging to solve optimally or exactly, but there are extremely efficient heuristics that find good approximate solutions. A naive approach would be to start by picking *K* random center points in the product rating space and assign every user to the nearest of these *K* points. This will give us an initial division of users into groups that will probably do terribly on the criterion we've specified (make within-group distances small compared to between-group distances). But now we can gradually make small adjustments to these *K* centers to improve the criterion until we get stuck and can't make any further improvement; in the language of algorithms, this would produce a locally optimal solution, but possibly not a globally optimal one. Once we have produced the groups or clusters, we can always produce a canonical user or type for each group—for instance, by averaging the ratings of the users in that group.

There are many machine learning algorithms that work much faster and better than this naive approach; they have rather technical names such as *K*-means, expectation-maximization, and low-rank matrix approximation. But for our purposes, they all share the same top-level goals: given a colossal and incomplete dataset of user product ratings or purchases (incomplete because virtually every user has purchased only a small fraction of Amazon products), discover a small number of canonical user types, which can then be used to make accurate recommendations of new products to users based on their type.

THE ECHO CHAMBER EQUILIBRIUM

So just like Waze, Amazon gathers all of our collective behavioral data (about shopping, instead of driving and traffic) and uses it to try to optimize recommendations for each of us individually (about what products to buy, instead of what route to drive). What's initially harder to see in this analogy is the sense in which shopping is a "game" the way commuting is, and whether Amazon is nudging its users toward a bad selfish equilibrium. Everyone else driving on the roadways clearly impacts you negatively in the form of traffic. But what consequences, good or bad, does everyone else's online shopping have for you?

From our description of collaborative filtering, it's at least clear there *are* consequences. When our collective data is used to estimate a small number of user types and you are assigned to one of them, the products suggested to you are narrowed by the scope of the model created, which is a function of everyone else's shopping activity. If the model implicitly learns that people who drive American cars and enjoy hunting are more likely to read Stephen King than postmodern fiction, they will be steered in that direction, even though they might have enjoyed Thomas Pynchon. Of course, people still have free will and can choose to ignore Amazon recommendations in favor of their own research and wanderings. But this is true of Google Maps and Waze as well—one can always choose the slow or scenic route instead of the time-saving selfish one, or opt out of using such apps at all. But the more of us that adopt their suggestions, the more our collective behavior is influenced or even determined by them.

Perhaps in the case of shopping, we can argue we are being nudged toward a good or at least neutral equilibrium—one in which we all benefit from the insights gleaned by machine learning from our aggregate purchases, in the form of individually optimized recommendations for new products. Since we're not all competing for a limited resource such as the capacity of the roadways, maybe all of us can simultaneously optimize without making anyone else worse off, which is unavoidable in commuting.

But we might feel differently about the same methods applied to a very similar problem, which is news filtering. Platforms such as Facebook also apply the same powerful machine learning techniques to build individual profiles of user interests based on collective data, and use these models to choose what appears in their News Feed. The aforementioned narrowing that can occur in this process has oftentimes been referred to as an "echo chamber" (or "filter bubble")—meaning that users are being shown articles and information that align with or echo their existing beliefs and prejudices. And what articles appear on a user's News Feed can further reinforce the existing behavior that led the algorithm to select them in the first place, resulting in another feedback loop.

In game-theoretic terminology, these services may have led us to a bad equilibrium in which politics and public discourse have become polarized, and which encourages all of us to become less informed about, and less tolerant of, opposing perspectives. We're all best-responding—at any given moment, we'd prefer to read an article confirming rather than challenging our beliefs—but perhaps as a society we have been led to a less healthy place by technology. This polarization has exacerbated the effectiveness of "fake news" inside online communities that are rarely exposed to dissenting information. There have also been widespread incidents of deliberate manipulations, such as bogus Facebook accounts—complete with user profiles seeded with specific identities, activities, and opinions—designed so that the fake users will have their posts show up in the News Feeds of particular online communities or echo chambers.

Note that the Amazon and Facebook equilibria are largely the same, and both are the direct result of algorithmic, model-based systems trying to simultaneously optimize everyone's choices. It's just that when an informed and deliberative society is at risk, the selfish equilibrium feels much worse than when we're all just choosing our next novel or vacation. As with fairness (Chapter 2), the stakes matter.

QUANTIFYING AND INJECTING DIVERSITY

If we don't like the echo chamber equilibrium for news filtering, or maybe even for shopping, how might we improve the algorithms used by platforms like Facebook and Amazon? One natural approach is to increase the diversity or exploration in the recommendations made to users—to deliberately expose people to news articles or products that differ from their obvious preferences and past behavior. Furthermore, this need not be done in a random or haphazard fashion (which might strike users as simply irrelevant and irritating); it can be done algorithmically.

Consider the mapping of users to types implemented by methods such as collaborative filtering. Because such models position users and types in space, there is also a quantitative measure of distance between them. So not only can the model estimate your type, it also knows how similar or different you are from all of the other types. For example, in our three-book example above, we can measure the distance between the centers of the circle cloud that likes *The Shining* and the postmodern fiction triangle cloud.

If we wanted news filtering algorithms that really challenged your worldview, we could deliberately pepper you with articles most aligned with your opposite type—the one farthest in space from your own. Of course, this might be too much too soon, and simply alienate or even offend users. But the point is that such algorithms can provide a "knob" (akin to the fairness and privacy parameters discussed in earlier chapters) that can be adjusted (even by individual users, if platforms so chose) from purely selfish echo chamber behavior, to making recommendations a bit outside your comfort zone, to really showing you how your polar opposites see the world. And of course if want to be algorithmically transparent, we can always choose to put the exploratory recommendations in a clearly marked "Opposing Viewpoints" sidebar.

So unlike our proposals for the commuting game—which required us to adopt different and more speculative algorithms (like Maxwell 2.0) rather than the selfish one in order to achieve socially better solutions—we can attempt to address the echo chamber equilibrium with algorithms and models that are only minor variants of the ones currently deployed. It might just involve changing a few lines of existing code.

MEDICAL MATCHMAKING

Throughout this chapter, we have deliberately chosen to illustrate the interplay between individual preferences, collective welfare, game theory, and algorithms through examples with which most of us have daily experience, such as driving, shopping, or reading news. But there are many more specialized settings in which algorithmic game theory has long played a central role in highly consequential decisions.

One such setting is broadly known as *matching markets* in economics. While this phrase may bring to mind dating apps like Coffee Meets Bagel, matching markets are usually found in much more formal scenarios in which we want to pair up individuals with each other, or individuals with institutions. One long-standing application area is medical residency hiring, in which the approach we'll describe is implemented as the National Resident Matching Program (NRMP, affectionately known as "The Match").

The basic problem formulation is as follows. Candidates for medical residencies each have an individual ranking of potential programs. For example, suppose the candidates Elaine and Saeed have the following ranked list of residencies (ignore the annotating characters for now, which we discuss shortly):

Elaine	**Saeed**
Harvard	*Cornell*
Johns Hopkins	*UC San Diego @*
UC San Diego &	*Harvard #*
Baylor	*Johns Hopkins*

Based on application materials and interviews, the schools of course also have their own ranked lists of candidates, such as:

Harvard	**UC San Diego**
Saeed #	*Roger*
Elaine	*Saeed @*
Roger	*Elaine &*
Gwyneth	*Mary*

So this is a two-sided market—candidates and hospitals—and there are also capacity constraints, because each candidate can of course take only one residency, and each hospital can only take a limited number of residents (which for simplicity we'll assume is also just one). Thus, as with dating and commuting, we once again have a large system (many thousands of applicants, and hundreds of schools) of interacting, competing preferences and would like to specify a desirable solution—and a fast algorithm for finding one.

Let's approach the notion of a desirable solution by first specifying what we *don't* want to happen. Consider candidates Elaine and Saeed in the example above. Suppose we match Elaine with UC San Diego (indicated by the "&" characters next to both), and Saeed with Harvard (indicated by the "#" characters next to both). Then regardless of any other matches we make, this solution is *unstable*, because Saeed prefers UC San Diego to Harvard, and UC San Diego prefers Saeed to Elaine—that is, the match indicated by the "@" characters would be preferable to both Saeed and UC San Diego, compared to their respective outcomes under the & and # matches. So while Elaine and Harvard might be happy, Saeed and UC San Diego have an incentive to deviate or defect from their assigned matches (Harvard and Elaine, respectively) and hook up with each other instead. A solution in which there are no such potential defections is called a stable matching. A matching with this property is not at risk of unraveling as students and hospitals iteratively defect from their proposed matches.

A stable matching is conceptually quite similar to a Nash equilibrium, but now two parties (a candidate and a medical school) must jointly defect to a mutually preferred outcome, due to the two-sided nature of the market. And like a Nash equilibrium, a stable matching in no way promises that everyone will be satisfied with the outcome: a candidate assigned to her 117th-favorite hospital may not be happy, but as in a Nash equilibrium, there is nothing she can do about it, because the 116 hospitals she prefers already have candidates they like better than her. Candidates and hospitals that are paired together in a stable matching are stuck with each other. Nevertheless, a stable matching is an intuitive solution to such pairing or assignment problems—certainly any solution that is *not* a stable matching is vulnerable to defections and is therefore problematic—and it is a *Pareto optimal* solution as well, in the sense that there is no way to make anyone better off without making someone else worse off (similar to the accuracy-fairness Pareto curves discussed in Chapter 2).

Like other topics in this chapter, stable matchings have both a long history and fast algorithms for computing them, going back at least to the seminal 1962 work of David Gale and Lloyd Shapley. The so-called Gale-Shapley algorithm is sufficiently simple that it even has a plain English description on Wikipedia, whimsically phrased as pairing men and women via traditional Victorian courtship:

- In the first round, *a*) each unengaged man proposes to the woman he prefers most, and then *b*) each woman replies "maybe" to her suitor that she most prefers and "no" to all other suitors. She is then provisionally "engaged" to the suitor she most prefers so far, and that suitor is likewise provisionally engaged to her.
- In each subsequent round, *a*) each unengaged man proposes to the most-preferred woman to whom he has not yet proposed (regardless of whether the woman is already engaged), and then *b*) each woman replies "maybe" if she is currently not engaged or if she prefers this suitor over her current provisional partner (in this case, she rejects her current provisional partner who becomes unengaged). The provisional nature of engagements preserves the right of an already-engaged woman to "trade up" (and, in the process, to "jilt" her until-then partner).
- This process is repeated until everyone is engaged.

The Gale-Shapley algorithm has two very nice properties. First, regardless of the preferences, everyone gets matched (if there are equal number of men and women, and they don't deem any of their potential partners as absolutely unacceptable; for instance, every medical student wants to do a residency no matter what). Second, the matching computed by the algorithm is stable in the sense we described above. Algorithms generalizing to the cases where there are unequal numbers of men and women (or students and medical schools), or where one side of the market can accept more than one partner (as in medical residencies), also exist. These algorithms are in widespread practical use, including in assigning actual medical residencies and other competitive admissions settings, such as matching students to public high schools and matching pledges to sororities in universities. (In contrast, US undergraduate college admissions are generally done in a much more haphazard fashion, giving rise to experiments in admissions office gamesmanship such as early decision, early action, requiring standardized tests or not, additional essays, and the like—all to the exhaustion and frustration of applicants and their parents.)

One of the most striking applications of algorithmic matching in the real world—one that literally has saved human lives—is to the problem of paired kidney donation. Many people with kidney disease die every year while awaiting a transplant donor, and the problem is exacerbated by the fact that the blood type of a donor must be compatible with that of the recipient to be biologically viable (there are also a variety of other medical compatibility constraints). We can view the blood type and biology of a donor as a form of "preferences" over recipients—a donor "prefers" to donate to compatible recipients, and not to incompatible ones. Similarly, a recipient prefers to receive a transplant from a compatible donor.

While there are many details that make this problem more complicated than medical residency matching, there are again practical, scalable algorithms that maximize the efficiency of the solution found, where here efficiency means maximizing the total number of compatible transplants that occur globally—ideally across all hospitals, not just within a single one. For his algorithmic and game-theoretic insights on this problem (and the others we have mentioned, including the medical residency match) and his efforts to convince the medical community and hospitals that it was worth the effort to pool their transplant donors, recipients and data, Alvin Roth was awarded the 2012 Nobel Prize in economics—along with the aforementioned Lloyd Shapley, whose early work initiated the era of algorithmic matching.

ALGORITHMIC MIND GAMES

We've now seen a variety of modern settings (and potentially future ones, such as self-driving cars) in which game-theoretic modeling can provide both conceptual guidance and algorithmic prescriptions to problems in which a large number of individuals or institutions have complex and potentially competing preferences. The proposed algorithms are socially aware in the sense that they attempt, albeit imperfectly, to mediate these preferences in a way that has socially desirable properties such as efficiency (e.g., low collective commute time), diversity (e.g., of news exposure), or stability (e.g., of matchings).

A rather different and more recent use of game theory is for the internal design of algorithms, rather than for managing the preferences of an external population of users. In these settings, there is no actual human game, such as commuting, that the algorithm is helping to solve. Rather, a game is played in the "mind" of the algorithm, for its own purposes.

An early example of this idea is self-play in machine learning for board games. Consider the problem of designing the best backgammon-playing computer program that you can. One approach would be to think very hard about backgammon strategy, probabilities, and the like, and hand-code rules dictating which move to make in any given board configuration. You would try to articulate and impart all of your backgammon wisdom in computer code.

A rather different approach would be to start with a program that knows the rules of backgammon (i.e., which moves are legal in a given configuration) but initially knows absolutely nothing about backgammon strategy (i.e. how to play well, rather than just legally). The initial version of this program might simply choose randomly among its legal moves at every round—surely a losing strategy against even a novice player. But if we make this initially naive program adaptive—that is, if its strategy is actually a model mapping board configurations to next moves that can be tuned and improved with experience—then we can take two copies of this program and have them improve by playing *each other*. In this way we turn playing backgammon into a machine learning problem, with the requisite data being provided by simulated self-play.

This simple but brilliant idea was first successfully applied by Gerry Tesauro of IBM Research in 1992, whose self-trained TD-Gammon program achieved a level of play comparable to the best humans in the world. (The "TD" stands for "temporal difference," a technical term referring to the complication that in games such as backgammon, feedback is delayed: you do not receive feedback about whether each individual move was good or bad, only whether you won the entire game or not.) In the intervening decades, simulated self-play has proven to be a powerful algorithmic technique for designing champion-level programs for a variety of games, including quite recently for Atari video games, and for the notoriously difficult and ancient game of Go.

It might seem natural that internal self-play is an effective design principle if your goal is to actually create a good game-playing program. A more surprising recent development is the use of self-play in algorithms whose outward goal has nothing to do with games at all. Consider the challenge of designing a computer program that can generate realistic but synthetic images of cats. (We'll return shortly to the question of why one might be interested in this goal, short of simply being a cat fanatic.) As with backgammon, one approach would be the knowledge-intensive one—we'd gather cat experts and study images of cats in order to understand their coloring, physiology, and poses, and try to somehow encode all of this expertise in a program that generated random, photo-realistic cat images. It seems difficult to even know where to begin.

Or we could again take the simulated self-play approach. The high-level idea is to invent a game between two players that we'll call Generator and Discriminator. The goal of Generator is to create or generate good artificial cat images. Discriminator is given a sample of fake cats created by Generator, as well as a large collection of real cat images, and has the goal of reliably discriminating between the fake and real cats. Discriminator can actually be a standard machine learning algorithm whose training data labels the real cats as positive examples and the fake ones as negative examples. But as with TD-Gammon, it is crucial that *both* players are adaptive.

At the start of the game, Generator (who does not even have the luxury of seeing any real cat images) plays terribly, creating pictures that look like random collections of pixels. Thus Discriminator's task is comically easy—just differentiating between real cats and garbled bits. But after the first round, once Discriminator has committed to its first model, Generator uses this model to modify its fake cats in a way that makes Discriminator slightly more confused about the real versus the fake . . . which in turn forces Discriminator to revise its model to solve this slightly harder problem. We continue this back-and-forth ad infinitum. If both players become experts at their respective tasks, Generator creates incredibly realistic cats, which the Discriminator might still be able to distinguish from the real thing better than humans could. But if the Generator is sufficiently good at learning, this is clearly a losing game for the Discriminator.

The technical name for the algorithmic framework we have been describing is a generative adversarial network (GAN), and the approach we've outlined above indeed seems to be highly effective: GANs are an important component of the collection of techniques known as deep learning, which has resulted in qualitative improvements in machine learning for image classification, speech recognition, automatic natural language translation, and many other fundamental problems. (The Turing Award, widely considered the Nobel Prize of computer science, was recently awarded to Yoshua Bengio, Geoffrey Hinton, and Yann LeCun for their pioneering contributions to deep learning.)

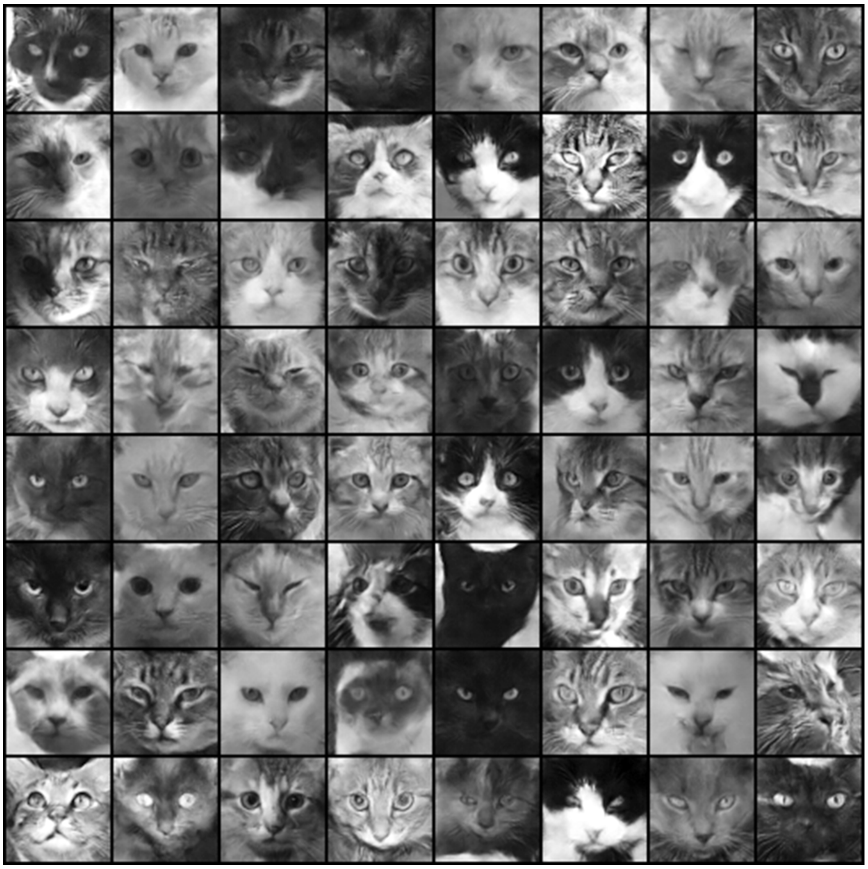


Fig. 21. Synthetic cat images created by a generative adversarial network (GAN), from https://ajolicoeur.wordpress.com/cats.

But with all of this discussion of simulated self-play and fake cats, it might seem like we have strayed far from the core topic of this book, which is the interaction between societal norms and values and algorithmic decision-making. However, recent research has shown that these same techniques can in fact play a central role in the design of better-behaved algorithms as well.

For example, recall our discussion of "fairness gerrymandering" back in Chapter 2—the phenomenon that despite building a model that does not discriminate by gender, race, age, disability or sexual orientation separately, we nevertheless end up with one that is unfair to gay disabled Hispanic women over age fifty-five making less than $50,000 a year. This was just another instance of machine learning not giving you "for free" something you didn't say you wanted. We mentioned briefly in that chapter that one solution to this problem involved an algorithm that simulated play between a Learner who would like to minimize error and a Regulator who continually confronts the Learner with subgroups suffering discrimination under the Learner's current model. This is another example of simulated game play as design principle, with the Regulator taking the place of a backgammon program or a Generator. Here the Regulator's goal (fairness) may be in conflict with the Learner's (accuracy), and the outcome (which is actually a Nash equilibrium of a precisely defined game) will be a compromise of the two—as desired.

Similarly, game-theoretic algorithm design has also proven useful in differential privacy. For example, while there may not be much motivation to generate fake cat pictures—we have plenty of the real thing—there is very good reason to generate realistic-looking but fake or synthetic medical records. In this case, the real thing can't generally be widely shared because of privacy concerns—often to the detriment of scientific research. One recent application of GANs is to the generation of highly realistic sets of medical records that can be made publicly available for research purposes while preserving the individual privacy of the patients whose real records trained the GAN. This is again achieved by framing the algorithm as a game—here between a Generator who desires to keep the synthetic dataset as close to the real dataset as possible and a Discriminator who wants to point out differences. So long as the Discriminator is differentially private, the synthetic data produced by the Generator will be as well. So while GANs are in their relative infancy, scientists are starting to find compelling (non-feline) uses for them.

GAMES SCIENTISTS PLAY (WITH DATA)

Most of this chapter examined settings in which people have preferences that may conflict with each other, and the ways in which algorithms can play a role in their mediation and management (for better or worse). Many of these "games" involved everyday activities such as driving or shopping, and some were more rarified, like assigning medical residencies.

We have one last case study in which the participants may not think of themselves as players in a complex game—but they are. It's the realm of modern scientific research, especially in the rapidly expanding disciplines in which data analysis and predictive modeling are playing an increasingly dominant role (which of course includes machine learning itself). The players in this game are the professors, graduate students, and industry researchers in data-driven fields. Their incentives involve the publication of novel and influential results, often requiring improvement in quantitative metrics such as the error rate on benchmark data sets or in the findings of prior experiments and analyses. Each new publication influences subsequent data-gathering, modeling, and choices made by the scientific community. It's a game where even individually careful scientists can participate in a bad equilibrium in which there is collective overfitting to widely used data sets—thus leading to spurious and false "scientific" findings.

It's a game that deserves a chapter of its own.