Dynamic Matching With Teams

Qingyun Wu*

Abstract

This note studies a dynamic matching model in which a social planner creates team based game sessions for sequentially arriving players and seeks a balance between fairness and waiting times. We derive a closed-form optimal matching policy and show that as the team size grows and the market becomes more balanced, greedy policies become less appealing. (JEL: C61, C78, D47)

Key words: dynamic matching; Markov decision process.

1 Introduction

Over the past 20 years the video game industry has been growing rapidly and its global revenue is estimated to be \$120.1 billion in 2019.[1] Many popular games have massive playerbases; for example, the number of monthly active players in Leauge of Legends is over 100 million, as of September 2016.[2] With such tremendous amount of gamers, hundreds of millions of game sessions are created every day. This note studies one issue that every player versus player (PVP) game is facing: matchmaking.

In competitive PVP games such as League of Legends, Dota 2, Overwatch, CS:GO, etc, creating high quality matches is quite a challenge. For these games, a game session consists of two teams playing against each other. And the foremost concern for the matchmaker is the trade-off between player waiting time and skill balance between the two teams.[3] Players dislike waiting and hope to be assigned into a game quickly.

*Department of Economics, Stanford University, Stanford, CA 94305 (email: wqy@stanford.edu).
I thank Itai Ashlagi, Junnan He, Fuhito Kojima, Jacob Leshno, Xiaocheng Li and Alvin Roth for helpful discussions.

[1]According to a report from Superdata: `https://www.superdataresearch.com/2019-year-in-review?itemId=v4s42dbcypw4ka6cq60m57t6v9nbdx`

[2]Based on an interview with the CEOs of Riot Games: `https://www.polygon.com/2016/9/13/12891656/the-past-present-and-future-of-league-of-legends-studio-riot-games`

[3]Sometimes role assignment is also important. As Riot Gortok (an employee of Riot Games, the developer and publisher of League of Legends) once commented: "The broadest explanation of the problem (matchmaking) is we are constantly trying to balance the trade-off between how long a player is in queue, how closely matched the game is, and how often you get the position you desire." See this reddit post for details: `https://www.reddit.com/r/leagueoflegends/comments/66i2q8/matchmaking_for_challengermaster_really_bugged_at/dgizsxb/`

However for the sake of fairness and ultimately player experience, it is often not a good idea to create game sessions whenever enough players have entered the queue. Many games use ELO[4] based rating systems to measure player skill. To make a game exciting and competitive, the matchmaking system needs to find players of similar ELOs for each game session and make the skill difference between the two teams as small as possible. Even though many games have concurrent players count close to, or even over 1 million (so market thickness appears not to be an issue here), finding a sweet spot between these two is no easy task, especially at the highest and lowest end of the skill spectrum.

Game companies often give top priority to fairness and competitiveness of the matching. Aside from the obvious benefit that even games are exciting and fun, it also protects new players: if rookies are matched with random players, they are likely to lose or even get stomped most of the time, which can be discouraging and therefore detrimental to the growth of playerbase. Furthermore, when matchmaking is mostly based on the time of entering the matching pool, it becomes easy to join the same game session as someone else (called "queue sniping"). This sometimes cause problems such as "stream sniping": nowadays a considerable amount of players stream on platforms like Twitch.tv and YouTube[5] and they are sometimes trolled by stream snipers (either their fans or malicious competitors), who can easily mess with streamers' gameplay as they observe streamers' positions and actions through the stream (called "ghosting"). A similar issue is called "teaming". In games with more than two teams/individuals that are supposed to work alone, e.g. PlayerUnknown's Battlegrounds, players can queue up with friends and effectively bypass the team size limit and create unfair advantage. With a skill based matching system, such issues become less prevalent.[6]

However, if one puts a lot of emphasis on fairness, queue time inevitably becomes an issue, especially at extremely high and low ELOs. In games like League of Legends, 10 players (5 vs 5) of similar skill level are needed for each game. Imagine now you are the rank 1 player in League of Legends, then the matchmaking system wants to match you with 9 other extremely skilled players. But it is highly likely that when you queue up for a game, there are not enough top players who are also looking for

[4]A commonly used rating system in real world sports such as Chess, Go, FIFA World Ranking, etc. See this wiki page for details: `https://en.wikipedia.org/wiki/Elo_rating_system`

[5]There are a lot of viewers: the League of Legends World Championship 2015 was watched by 334 million people, with an average of 4.2 million tuned in at the same time.

[6]Indeed it is hard to completely shut down queue sniping. In China, there is an ELO boosting business called "acting". The "director" and "actor" queue up for the same game, but on opposite sides, so the actor intentionally loses the game and the director gains ELO. It is called acting since if the actor goes straight up feeding he would get banned. So he has to act like he is trying to win, but the ultimate goal is to lose. Of course this is hard to pull off, since they would need two accounts with very close ELO ratings in the first place; and typically this only happens at very high ELO. In Dota 2 there is a similar story: one player (mallljK) exploited a flaw in the matchmaking system so that every game he played consisted of 10 of his own accounts and hit 10k solo ELO (800 above the second place).

a game, so you have to wait. In fact the queue time for high ELO players are often around 20-50 (with a single game lasts on average about 30) minutes. A Brazilian player, Jowsss, who was rank 1 in 3 vs 3 mode once waited 30 hours before he was assigned to a game, and he failed to accept since he was sleeping when the queue pops.[7]

This paper sets up a stylized dynamic matching model and characterizes the optimal matching policy. The literature on dynamic matching is fast growing. Baccara, Lee and Yariv (2020) is perhaps the closest to our paper. They study a two-sided dynamic matching market in which one square and one round, each with two possible types arrive in each time period and show that the optimal mechanism cumulates a stock of incongruent pairs up to a threshold and matches congruent pairs instantaneously. Leshno (2019) studies a dynamic matching problem with an overloaded waiting list and characterizes the optimal way of assigning priorities to the positions of the waiting list. Akbarpour, Li and Gharan (2020) models a kidney exchange market in which the dynamically arriving agents might perish if not matched soon enough. Their main insight is that waiting to thicken the market performs significantly better than greedily matching agents upon arrival. On the other hand, with a different model, but still on kidney exchange, Ashlagi, Jaillet and Manshadi (2013) finds that the benefit of matching in batches over the greedy policy is small.

2 Model

Let there be a discrete time horizon $t = 1, 2, 3, \ldots$ At each time t, a new player joins the matching pool and a matchmaker then decides whether/how to create game sessions each consisting of two n-player teams from the players currently waiting in the pool (if a game is formed, then those $2n$ players leave the matching system). Every player has a skill type known to the matchmaker.[8] In this note we study a simple setting in which there are only two skill types, high and low, denoted by H and L respectively. The probability of a H type arriving is q and the probability of a L type arriving is $1-q$. Of course, high and low are relative measures and therefore q is a parameter that depends on the goal of the matchmaker. For example, if creating a good environment for professional players is of high priority, then q is small; while if protecting new players is important, then q is large.

There are two types of costs.

Firstly, the matchmaker wants to create fair games: in each game, he aims to have an equal number of H players in both teams. Of course, the ideal situation is to always create game sessions consisting of a single skill type, which could take a long time, or even is impossible in game modes such that friends of different skill levels are

[7] Here is his reddit post with a screenshot `https://www.reddit.com/r/leagueoflegends/comments/6f8aap/what_is_the_longest_queue_time_ever/`

[8] It is typically true that the matchmaker knows players' skill levels through past win/loss records, often in the form of ELO ratings.

allowed to queue up together. Therefore matchmakers often settle for a weaker notion of fairness like the one defined here.[9] More specifically, whenever an imbalanced game is formed, there is a cost associated with that game equals to $\alpha \times I \times 2n$ (i.e. $\alpha \times I$ per player), where

$$I = |\text{number of H types in team } 1 - \text{number of H types in team } 2|,$$

and $\alpha \geq 0$ is a punishment parameter.

Secondly, as players dislike waiting, there is also a waiting cost. We assume a linear cost structure and normalize the waiting cost to be 1 per period per player.

Let C_t denote the total costs up until time t, which is the sum of the imbalance cost of the games formed at time $\leq t$ plus the total waiting costs occurred $\leq$ time t (If a player enters at time a and leaves at time b, $a \leq t \leq b$, then he contributes a waiting cost of $t - a$ in C_t). The benevolent social planner then would like to find the optimal matching policy that minimizes the expected time average of total costs, i.e. $\lim_{t\to\infty} \frac{E(C_t)}{t}$. Unfortunately with history dependent policies, this limit may not exist, in which case we shall replace the limit with liminf. For technical simplicity, we assume that the maximum size of the matching pool is finite, i.e. if there are $\geq M$ (for some arbitrarily large M) players waiting in the system, then at least one game session has to be formed. Under this mild assumption, the limit always exists under any stationary policy and an optimal stationary policy is guaranteed to exist. In other words, there exists a stationary matching rule $\mathcal{R}$ such that for any other matching rule $\mathcal{R}'$, $\lim_{t\to\infty} \frac{E(C_t^{\mathcal{R}})}{t} \leq \liminf_{t\to\infty} \frac{E(C_t^{\mathcal{R}'})}{t}$, see Chapter 8 and 9 of Puterman (2014). Now the problem is well-defined and we derive the optimal stationary policy in the next section.

3 Optimal Policy

The first thing to notice is that, in the optimal policy, I is at most 1, since otherwise we can exchange 1 H player from the team with more H's and 1 L player from the team with less H's, and obtain a match with less imbalance cost. Furthermore, we have the following lemma:

Lemma 3.1. *There exists an optimal stationary policy that forms any zero imbalance cost match immediately.*

Let's denote a sequence of realized arriving H and L types as an outcome path. Two remarks before we prove this lemma:

(I). Since our objective is to minimize the expected cost, the optimal policy may not be optimal along every outcome path. In fact it is possible that an optimal policy is never optimal for any realized outcome path. Consider the following example:

[9] Or more realistically, they may want both teams to have similar average ELOs.

suppose we take an action among $\{A, B, C\}$ and the payoffs of the actions depend on the outcome of a coin flip. If the coin comes up "Heads", the payoffs of A, B, and C are 10, 8, 1; while if the coin comes up "Tails", the payoffs are 1, 8, 10. Then overall action B is optimal in expectation, but for any realized coin flip, B is never the optimal choice.

(II). We should heed the timing of making a decision. Suppose players A, B, C, D enter the matching pool sequentially and originally A is matched to B and C is matched to D. Imagine we can improve matching qualities by switching these 2 matches, i.e. by matching A to C and B to D. However, if we want to switch, we can not make such a decision at the time when D arrives; instead, we have to decide at the time when we originally match A to B, before D enters our matching pool.

Proof of Lemma 3.1: First notice that whenever there are at least $2n+1$ players in the system, we can pick $2n$ of them and form a zero cost match. Suppose there is an optimal matching rule $\mathcal{R}$ that does not form a zero imbalance cost match consists of players $A_1, A_2, ..., A_{2n}$ (labeled according to time of arrival) immediately. Then there exists an outcome path P such that $A_1, A_2, ..., A_{2n}$ stay in the matching pool for at least 1 period, denote $\bar{t}$ as the first time which this phenomenon occurs, i.e. $\bar{t}$ is the time when A_{2n} arrives. We now construct a new policy $\mathcal{R}'$ that is weakly better than $\mathcal{R}$ on every path. For any outcome path P' that disagrees with P at some time $t \le \bar{t}$, this new matching rule agrees with $\mathcal{R}$ on P'. Now, suppose an outcome path P' agrees with P for all time $t \le \bar{t}$, we construct a new matching rule for each such path. First we match $A_1, A_2, ..., A_{2n}$ immediately at time $\bar{t}$ on path P'; however, we pretend that we did not create such a match and keep following $\mathcal{R}$ until the time when the first player(s) among $A_1, A_2, ..., A_{2n}$ is supposed to be matched according to $\mathcal{R}$. At this time we do nothing, but keep those players who are supposed to be matched by now in $\mathcal{R}$ but still not matched in $\mathcal{R}'$ in an imaginary matching bank (of course they still stay in the matching pool). Note at this point the size of the matching pools under $\mathcal{R}$ and $\mathcal{R}'$ is the same, although the composition may be different. We then pretend we followed $\mathcal{R}$ until this time, and keep following $\mathcal{R}$ until the next player(s) among $A_1, A_2, ..., A_{2n}$ is supposed to be matched according to $\mathcal{R}$. Now we put these non-A_i players in this match into our imaginary matching bank. Two cases: (I). If the matching bank has exactly $2n$ players, then we form a match, and our modification of $\mathcal{R}$ on this path P' ends: notice at this time the matching pools under $\mathcal{R}$ and $\mathcal{R}'$ are exactly the same, then we can follow $\mathcal{R}$ afterwards. (II). There are at least $2n+1$ players in the matching bank. We can pick $2n$ of them and form a zero cost matching, remove these players from our matching bank (and matching pool), and then pretend we followed $\mathcal{R}$ until now, and keep following $\mathcal{R}$ until the next player(s) among $A_1, A_2, ..., A_{2n}$ is supposed to be matched according to $\mathcal{R}$. We repeat this process until our matching bank is empty, at which point the matching pools under $\mathcal{R}$ and $\mathcal{R}'$ are exactly the same, then we can follow $\mathcal{R}$ afterwards. Notice comparing $\mathcal{R}$ and $\mathcal{R}'$, only the matches involving $A_1, A_2, ..., A_{2n}$ in $\mathcal{R}$ are changed (either in match composition or match time) (on each path P'). And the modified matches in $\mathcal{R}'$ are all guaranteed to be zero cost, except the last one. If the last match in $\mathcal{R}'$ is also zero cost, then clearly $\mathcal{R}'$ produces a weakly smaller imbalance cost than $\mathcal{R}$. If the last match in $\mathcal{R}'$ is imbalanced and incurs a cost $\alpha \times 2n$, then the total number of H types in these modified matches must be odd, and thus at least one of the matches under $\mathcal{R}$ must be imbalanced, i.e. the total imbalance cost among the modified matches in $\mathcal{R}$ must be at least $\alpha \times 2n$. Therefore in this case, $\mathcal{R}'$ also produces a weakly smaller imbalance cost than $\mathcal{R}$. Then no matter what, $\mathcal{R}'$ does weakly better than $\mathcal{R}$ in terms of imbalance cost. On the other hand, $\mathcal{R}'$ is also better than $\mathcal{R}$ in terms of waiting cost, since we form the first match involving $A_1, A_2, ..., A_{2n}$ earlier in $\mathcal{R}'$ than in $\mathcal{R}$, and all other match creation times are the same under both policies. Therefore if $\mathcal{R}$ is optimal, $\mathcal{R}'$ must also be optimal. However if it is optimal to form a zero cost match immediately at time $\bar{t}$, then by the optimality of stationary policies, it is also optimal to do so at every decision time.[10] □

With Lemma 3.1, we shall focus on stationary policies that form any zero imbalance cost match immediately.

Let's consider a Markov chain X_t that describe the state of the matching pool right before the t-th player joins the system. Then a generic element of the state space can be represented as (u, v), where u is the number of H types in the pool and v is the number of L types in the pool. When there are less than $2n$ players in the matching pool, we can do nothing but wait. When the matching pool contains $2n$ players, there are two situations ($u + v = 2n$ implies u and v must have the same parity): (I). u and v are both even. In this case we can create a zero cost match with these $2n$ players, and we shall do so immediately. (II). u and v are both odd. We have two options: the first one is to immediately match the players, incurring an imbalance cost α for each player, but no additional waiting costs, let's call it the **greedy policy**; the second one is to wait for one more period. Notice by Lemma 3.1, in the next period, if 1 H arrives, then we immediately form a zero cost match with $u + 1$ H's and $v - 1$ L's, and leave 1 L player in the matching pool; if 1 L arrives, then we immediately form a zero cost match with $u - 1$ H's and $v + 1$ L's, and leave 1 H player in the matching pool. This indicates that the specific values of u and v do not matter, only their parities do. Let's call this policy the **patient policy**. The only work left is to determine which of the greedy or patient policy is optimal for each parameter value α and q. This is not as easy as it appears to be, since the size of the state space of the Markov chain induced by the patient policy is $2n(n + 1)$, and tracking the stationary distribution of each state becomes tedious as n grows. However, we still have to begin our analysis by characterizing the properties of the steady state distribution of this Markov Chain.

Fix n, the state space of the Markov chain induced by the patient policy is the set $S = \{(u, v) | u+v \leq 2n-1 \textit{ or } u+v = 2n \text{ and u,v are odd}\}$, and the transition probabil-

[10] The proof strategy used here is similar to the one in Tsitsiklis (1994).

ities are the following: when $u+v \leq 2n-2$, $P_{(u,v)\to(u+1,v)} = q$ and $P_{(u,v)\to(u,v+1)} = 1-q$; when $u+v = 2n-1$, u odd, v even, $P_{(u,v)\to(0,0)} = q$, $P_{(u,v)\to(u,v+1)} = 1-q$; when $u+v = 2n-1$, u even, v odd, $P_{(u,v)\to(u+1,v)} = q$, $P_{(u,v)\to(0,0)} = 1-q$; when $u+v = 2n$, u, v odd, $P_{(u,v)\to(0,1)} = q$, $P_{(u,v)\to(1,0)} = 1-q$. Notice this is an irreducible and aperiodic Markov chain, thus there exists a stationary distribution, denote it by $\pi_{(u,v)}$. Also, define $\Pi_k = \sum_{u+v=k,\ (u,v)\in S} \pi_{(u,v)}$, for $0 \leq k \leq 2n$, i.e. Π_k is the proportion of time spent at a state in which k agents are in the matching pool in the steady state.

Lemma 3.2. *In the steady state distribution,* $\pi_{(0,1)} = \pi_{(1,0)} = \pi_{(0,0)}$.

Proof of Lemma 3.2: we first show $\pi_{(0,1)} = \pi_{(1,0)}$. Let δ_1 denote the probability that an outcome path starts from $(0,1)$ and ends at a state (u,v), where $u+v = 2n$ and u, v are odd, in $2n-1$ steps (the other possibility is that u and v are even). It is easy to compute that $\delta_1 = \sum_{k=1}^{n} \binom{2n-1}{2k-1} q^{2k-1}(1-q)^{2n-2k}$. Let δ_2 denote the probability that an outcome path starts from $(1,0)$ and ends at a state (u,v), where $u+v = 2n$ and u, v are odd, in $2n-1$ steps. We can compute that $\delta_2 = \sum_{k=0}^{n-1} \binom{2n-1}{2k} q^{2k}(1-q)^{2n-2k-1}$. Then $\delta_1 + \delta_2 = (q+(1-q))^{2n-1} = 1$. Let's write down the balance equation for $\pi_{(0,1)}$: the state $(0,1)$ can be reached from $(0,0)$ with transition probability $1-q$, and from a state (u,v), where $u+v = 2n$ and u, v are odd, with transition probability q. Then $\pi_{(0,1)} = (1-q)\pi_{(0,0)} + q\Pi_{2n}$. Since $\Pi_{2n} = \pi_{(0,1)}\delta_1 + \pi_{(1,0)}\delta_2$, we have

$$\pi_{(0,1)} = (1-q)\pi_{(0,0)} + q(\pi_{(0,1)}\delta_1 + \pi_{(1,0)}\delta_2)$$

similarly:

$$\pi_{(1,0)} = q\pi_{(0,0)} + (1-q)(\pi_{(0,1)}\delta_1 + \pi_{(1,0)}\delta_2)$$

cancel $\pi_{(0,0)}$ out in the above two equations, we get

$$\pi_{(0,1)}[q - \delta_1(2q-1)] = \pi_{(1,0)}[1 - q + \delta_2(2q-1)]$$

notice $1 - q + \delta_2(2q-1) = 1 - q + (1-\delta_1)(2q-1) = 1 - q + 2q - 1 - \delta_1(2q-1) = q - \delta_1(2q-1) > 0$, then $\pi_{(0,1)} = \pi_{(1,0)}$, plug this back in the first equation, we have $\pi_{(0,1)} = \pi_{(1,0)} = \pi_{(0,0)}$. □

At this point it is still a tedious task to compute the exact stationary distribution of this Markov chain. Luckily we only care about the expected waiting cost, which can be computed through Π_k. From Lemma 3.2 we have $\Pi_0 = \pi_{(0,0)} = 1/2(\pi_{(0,1)} + \pi_{(1,0)}) = 1/2\Pi_1$. It is also clear that $\Pi_1 = \Pi_2 = ... = \Pi_{2n-1}$, since any state with k elements is followed by a state with $k+1$ elements; and the only way to get to a state with $k+1$ elements is through a state with k elements, for $k = 1, 2, ..., 2n-2$. Finally $\Pi_{2n} = \pi_{(0,1)}\delta_1 + \pi_{(1,0)}\delta_2$ implies $\Pi_{2n} = 1/2\Pi_1$. Therefore $2\Pi_0 = \Pi_1 = \Pi_2 = ... = \Pi_{2n-1} = 2\Pi_{2n}$, combine this with $\sum_{k=0}^{2n} \Pi_k = 1$, we have $\Pi_0 = \Pi_{2n} = \frac{1}{4n}$, and $\Pi_1 = \Pi_2 = ... = \Pi_{2n-1} = \frac{1}{2n}$.

Now we can compute the expected waiting cost per period under the patient policy, which is

$$\frac{1}{4n} \times 0 + \frac{1}{2n} \times (1 + 2 + 3 + 4 + ... + 2n - 1) + \frac{1}{4n} \times 2n = n.$$

Notice patient policy produces no imbalance cost, therefore its total expected matching cost per period is n.

Next we compute the expected matching cost of the greedy policy. Every $2n$ periods the greedy policy forms a match, and the total waiting cost in these $2n$ periods is $1 + 2 + 3 + ... + 2n - 1 = n(2n - 1)$ and the expected imbalance cost is $2n\alpha \sum_{k=1}^{n} \binom{2n}{2k-1} q^{2k-1}(1-q)^{2n-2k+1} = 2n\alpha\frac{1-(2q-1)^{2n}}{2}$. Thus the total expected matching cost per period for the greedy policy is $\frac{1}{2n}(n(2n-1) + 2n\alpha\frac{1-(2q-1)^{2n}}{2}) = n - 1/2 + \alpha\frac{1-(2q-1)^{2n}}{2}$.

Therefore greedy is better than patient if and only if $n - 1/2 + \alpha\frac{1-(2q-1)^{2n}}{2} \leq n$, i.e. when $\alpha \leq \frac{1}{1-(2q-1)^{2n}}$. To summarize:

Theorem 3.3.

When $\alpha \leq \frac{1}{1-(2q-1)^{2n}}$, the greedy policy is optimal, i.e. we form a match whenever we have $2n$ players in the matching pool.

When $\alpha \geq \frac{1}{1-(2q-1)^{2n}}$, the patient policy is optimal, i.e. when there are $2n$ players in the matching pool, if they form a zero imbalance cost match, create such a match; otherwise wait for one period and form a zero imbalance cost match out of the $2n+1$ players.

This result has three major implications: First, (the obvious one) when α is small, greedy is more appealing. Second, greedy is more likely to beat patient when q is away from $1/2$: notice the total waiting cost is independent of q for both algorithms, while the greedy algorithm is less likely to create an imbalanced matching when q moves away from $1/2$. Third, if $q \neq 1/2$, then the greedy policy becomes less appealing as n grows large.

References

[1] Akbarpour, Mohammad, Shengwu Li, and Shayan Oveis Gharan. “Thickness and information in dynamic matching markets.” Journal of Political Economy 128.3 (2020): 783-815.

[2] Ashlagi, Itai, Patrick Jaillet, and Vahideh H. Manshadi. ”Kidney exchange in dynamic sparse heterogenous pools.” arXiv preprint arXiv:1301.3509 (2013).

[3] Baccara, Mariagiovanna, SangMok Lee, and Leeat Yariv. “Optimal dynamic matching.” Theoretical Economics 15.3 (2020): 1221-1278.

[4] Leshno, Jacob. “Dynamic matching in overloaded waiting lists.” Available at SSRN 2967011 (2019).

[5] Puterman, Martin L. Markov decision processes: discrete stochastic dynamic programming. John Wiley & Sons, 2014.

[6] Tsitsiklis, John N. “A short proof of the Gittins index theorem.” The Annals of Applied Probability (1994): 194-199.