Dynamic Weighted Matching with Heterogeneous Arrival and Departure Rates

Natalie Collina[1]*, Nicole Immorlica[2], Kevin Leyton-Brown[3], Brendan Lucier[2], and Neil Newman[3]

[1] Harvard University, nataliecollina@gmail.com
[2] Microsoft Research, {nicimm,brlucier}@microsoft.com
[3] University of British Columbia, {kevinlb,newmanne}@cs.ubc.ca

Abstract. We study a dynamic non-bipartite matching problem. There is a fixed set of agent types, and agents of a given type arrive and depart according to type-specific Poisson processes. The value of a match is determined by the types of the matched agents. We present an online algorithm that is (1/8)-competitive with respect to the value of the optimal-in-hindsight policy, for arbitrary weighted graphs. This is the first result to achieve a constant competitive ratio when both arrivals and departures are random and unannounced. Our algorithm treats agents heterogeneously, interpolating between immediate and delayed matching in order to thicken the market while still matching valuable agents opportunistically.

1 Introduction

Matching markets are ubiquitous in online platforms. Sponsored search auctions like Google Adwords match ads and users, ridesharing systems like Uber and Lyft match drivers and riders, online markets like Amazon and eBay match sellers and buyers. In each case, the value of a match is a function of the types of participating agents. In sponsored search auctions, a restaurant ad is more valuable when matched to a geographically co-located user. In ridesharing systems, a driver and rider have higher utility for being matched to each other if they are nearby. In an online market, buyers might have heterogeneous preferences over service/product quality and price trade-offs which impact match quality.

The role of the platform is to find high-value matches. However, this task is significantly complicated by the fact that agents arrive and depart dynamically over time, and may fail to inform the platform of their departure. In this paper, we mitigate this complication by assuming that agents have known Poisson arrival and departure rates that are a function only of their type. This allows us to characterize the optimal expected value from matches using a linear program. This program bounds the rate at which each pair of types match to one-another in the optimal solution. Our algorithm uses these LP-based estimates of the optimal rates as guidelines. When an agent arrives to the market, we attempt to match it to each previous agent with a probability equal to a scaled-back version of the corresponding rate. We prove the resulting algorithm is a constant approximation to the optimal-in-hindsight policy, with competitive ratio at most 8. While we motivate our problem in the context of bipartite matchings, we note our solution holds for general non-bipartite graphs.

There is a significant body of prior literature on dynamic stochastic matching in settings where agent departures are immediate or deterministic (and hence predictable) [9, 11, 15, 12, 13], or where the platform is informed immediately before an agent departs [1, 5, 8, 19]. In such settings, it is natural for the platform to delay matches until an agent is about to depart, in order to maximize the set of available options. In contrast, when the platform cannot predict departures, there is a tension between taking a guaranteed (but potentially suboptimal) match now, or pushing one's luck to see if a better match arrives later. The main technical challenge in developing an online policy is navigating this tradeoff for agents of different types.

Our LP-based approach is certainly not new in the context of stochastic matching, but we find that our result has several interesting qualitative insights, especially for settings where agent departures are random, heterogeneous, and unannounced. First, our algorithm treats matches heterogeneously. For some matches, the linear program suggests forming them at a high rate. Our algorithm treats these matches as a greedy algorithm would, matching them (almost) immediately upon arrival. For other matches, the linear program suggests forming them at a low rate. Our algorithm treats these matches more like a periodic clearing algorithm would, allowing the market to thicken before attempting the matches.

This heterogeneous treatment is important for good approximations in our setting. Consider, for example, an environment with two types of buyers, low and high, and one type of seller. The low buyers arrive frequently to the market and depart at a constant rate, whereas the sellers arrive less often. The high buyers arrive much less frequently than the sellers, and depart immediately after they arrive, but matches involving these high buyers account for almost all the value of the optimal policy. In this case, it is important to greedily match the high buyers and delay matches with the low buyers to thicken the market. A uniformly greedy policy, that immediately matches all agents, will likely have no sellers in the market when high buyers arrive, as there are always low buyers available to match with them. A periodic clearing algorithm that attempts to thicken the market by delaying all matches for a fixed period of time will likely have no access to high buyers at match time, since high buyers depart immediately after they arrive. See the full version of the paper for a more detailed description of such an example.

Another qualitative insight of our result is the importance of being conservative in matching attempts. Our algorithm scales back the match-rate estimates of the linear program by 50%. At first blush, this might seem incredibly wasteful. However, this scaling is provably necessary: we show in the full version of the paper that if the algorithm does not perform this scaling then it cannot achieve any bounded approximation to the optimal matching. Intuitively, the issue is that the matching policy must leave some slack in the system — by leaving a certain fraction of agents unmatched — in order to take advantage of unexpected fortuitous events where a very valuable match becomes possible. Since an optimal LP solution typically would leave no such slack, one can instead guarantee it by being conservative when matching.

As is common in the dynamic stochastic matching literature, our approach is to solve an LP relaxation of the offline optimal matching problem, then use this solution as guidance for our online matching policy. We prove that the resulting policy obtains a constant approximation to the LP benchmark, which is only stronger than the offline optimal match value (and hence the optimal online policy). The main technical hurdle is that the outcome of matching attempts is determined by the state of which types of agents are present in the market, and this introduces correlations across time. For instance, whether a certain type of agent is present in the market is (negatively) correlated with the presence of other agents that generate high value from matches with it. In principle, such correlations could result in scenarios where a certain type is either not present at all or is overabundant, impeding our ability to match the LP relaxation which assumes smoothness across time. We address this issue by bounding the impact of such correlations, by coupling the availability of agents in the system with Poisson processes that dominate (or are dominated by) them.

1.1 Related Literature

There is a vast recent literature on algorithms for online matching (sometimes called online task arrival). In a seminal paper, Karp et al. [14] consider an (unweighted) online bipartite matching problem where one side of the graph is static and the vertices of the other side arrive online. They show that a randomized greedy matching method obtains a $(1-1/e)$ approximation and that this is tight. This was later extended by Mehta et al. [17] to a generalized weighted matching environment motivated by ad auctions, with budget constraints on the static side of the market. Both of these results assume adversarial types.

Stochastic variants of the online bipartite matching problem have been studied as well. Feldman et al. [9] consider a stochastic variant in which vertex types on the online side of the market are drawn i.i.d. from a fixed distribution. They showed how to beat the adversarial bound of $(1-1/e)$ in this stochastic setting, using an LP-based approach that solves for a fractional (expected) matching, then rounds online using a flow decomposition. This led to a sequence of papers that improved the approximation factors for both the weighted and unweighted versions of the stochastic problem [11, 15], including variants with stochastic rewards [16, 18] and with capacities on the fixed side [2]. Gravin and Wang [10] obtain a constant approximation for a related variant inspired by prophet inequalities, where edges (rather than nodes) arrive online and must be matched immediately or lost.

Our model is closer in spirit to the literature on dynamic matching, where agents on both sides of the market arrive and depart over time. An algorithm proposes matches online between agents that are simultaneously present. Huang et. al [12] study an unweighted model in which node arrivals and departures are adversarial, but nodes announce when they are about to depart. They derive constant competitive online algorithms; in a later paper, Huang et al. [13] find tight competitive ratios. Akbarpour et al. [1] similarly consider an unweighted version in which agents depart at arbitrary times and inform the market when they are about to depart, but arrivals are stochastic. In this case, it is approximately optimal to match agents as they go critical. On the other hand, they show that without departure warnings, greedily matching agents as they arrive is nearly optimal. As the graph is unweighted in their model and agents are homogenous, analysis can proceed by studying the limiting distribution of the number of agents in the market.

The case of weighted matching with departure warnings was studied by Ashlagi et al. [5], and they obtain a constant approximation to the optimal weighted matching. When agents on both sides arrive according to a known IID random process, Dickerson et al. [8] provide constant competitive algorithms under the assumption that one side (say workers) never depart until they are assigned, and the other side (say tasks) depart immediately after arrival if unassigned. Truong and Wang [19] consider a related weighted bipartite matching model where agents arrive according to a general stochastic process, agents on one side depart after a fixed deterministic amount of time, agents on the other side depart immediately after arrival if unassigned, and they likewise obtain constant competitive algorithms. Importantly, in all of these works it is assumed that the platform knows when an agent is about to leave the system, either because this can be perfectly predicted or because the platform is explicitly notified, and the platform can therefore wait until an agent "goes critical" before attempting a match. In contrast to these works, we assume the platform is not notified of (and cannot predict) impending departures.

Independently and concurrently with our work, Aouad and Saritac [4] studied a similar model of dynamic matching with unannounced departures. They likewise find that there is a tension between greedy matching and batching. They develop an online algorithm guided by a quadratic program, and show that it is $(4e/(e-1))$-competitive for arbitrary compatibility graphs. In contrast, our method is based on linear programming (rather than quadratic programming), and our competitive ratio bound is weaker (8 versus $4e/(e-1)$). They also study a cost-minimization version of the problem, for which they develop an online algorithm that they analyze theoretically and evaluate on empirical data. We leave open the question of whether a combination of the ideas in these works could be used to develop algorithms with improved competitive ratio.

Other papers consider the related problem of minimizing average waiting time. Anderson et al. [3] find that matching agents as they arrive is nearly optimal even with departure warnings. Ashlagi et al. [6] consider a model with two agent types – hard-to-match and easy-to-match – and derive structural insights about policies that miminize average waiting time. Baccara et al. [7] consider a hybrid model with two agent types in which agents have varying match values and also incur waiting costs (but never leave the system).

2 Preliminaries

We consider a model with agents that arrive and depart over time. The type space of agents is X. Agents of type $x \in X$ arrive according to a Poisson point process of rate λ_x.[4] Each agent of type x that arrives then departs at Poisson rate μ_x. For an agent i of type x, we will write a_i and d_i for its realized arrival and departure times, respectively. Throughout, we refer to types of agents with letters x and y, and to specific agents with letters i and j.

A *matching* is a set τ of times and a pair of matched agents for each time $t \in \tau$. A matching is *feasible* if, for all matching times $t \in \tau$, the agents matched at t a) have already arrived and not yet departed, and b) have not been matched to anyone else at or before time t. The value of matching an agent of type $x \in X$ to an agent of type $y \in X$ is v_{xy}. For convenience, we sometimes denote the total value of all matches made at time t by v_t.

A *matching policy* chooses, at each time t, based only on the history up until time t, whether to match a pair of agents or to make no match. A *policy with hindsight* can revise past decisions, whereas for an *online policy*, all decisions are irrevocable. For any policy and time T, let $\tau(T)$ be all times $t \le T$ at which it made a match,[5] and v_t be the value of the matches made at time t, if any. Then the value of the policy is:

$$\liminf_{T \to \infty} \frac{1}{T} \cdot E\left[\sum_{t \in \tau(T)} v_t\right]$$

where the expectation is over the randomness in the arrival/departure process as well as any randomness in the policy. That is, the policy's value is the long-run average value of matches made per unit of time.

2.1 Poisson Processes

We now describe Poisson processes more formally. A point process is a random countable set of points $Z = \{z_1, z_2, \dots\}$. We restrict attention to the case $Z \subset R_{\ge 0}$, where we can interpret Z as a collection of event times. We refer to a point process by its set of points Z, which we think of as a random variable.

For any $T \ge 0$, we'll write $n_Z(T)$ for the number of points in $Z \cap [0, T]$; we think of this as the (random) number of events that occur before time T. Given two point processes Z and Y, we'll say that Z stochastically dominates Y if there is a coupling between Z and Y such that, for each $T > 0$, $\Pr[Y \subseteq Z] = 1$.

A static Poisson point process of rate $\lambda > 0$ is a point process such that

1. the set of points in any two disjoint intervals are independent, and

[4] We discuss Poisson processes more formally in Section 2.1.

[5] Note for an online policy, $\tau(T) \subseteq \tau(T')$ whenever $T \le T'$; however this need not hold for a policy with hindsight.

2. the number of points in any given interval of length T follows a Poisson random variable with parameter (mean) λT.

From this point onward we'll refer to static Poisson point processes as just Poisson processes, for convenience. The following standard facts about Poisson processes will be helpful in our analysis.

Fact 1 *Given a Poisson process Z of rate λ, write $n_Z(T)$ for the number of events that occur before time T. Then $E[n_Z(T)] = T\lambda$. Moreover, $\lim_{T\to\infty} n_Z(T)/T$ exists and equals λ with probability 1.*

Fact 2 *Suppose we have Poisson processes $Z_1, \ldots, Z_n$ of rates $\lambda_1, \ldots, \lambda_n$ respectively. Then the probability that the earliest event (i.e., minimum point) in $\cup Z_i$ lies in Z_i is $\lambda_i/(\sum_k \lambda_k)$.*

Fact 3 *Suppose Z is a Poisson process of rate λ, and Z' is a random set generated by adding each $z \in Z$ to Z' independently with probability p. Then Z' is a Poisson process of rate λp.*

A corollary of Fact 3 is that if Z is a Poisson process of rate λ and Z' is a Poisson process of rate $\lambda' < \lambda$, then Z stochastically dominates Z'. This is because we can couple Z and Z' by first realizing Z, then adding each element of Z to Z' independently with probability λ'/λ.

3 An Upper Bound

We construct an online policy whose value is a constant fraction of the optimal-in-hindsight policy. To do so, we develop a linear-programming upper bound on the value of the optimal-in-hindsight policy for large time horizons.[6] The value of the optimal solution is the expectation over the randomness in arrivals and departures of instance-optimal solutions, and so can be written as the expectation of the sum of match values.

In the following LP, the variable α_{xy} is the fraction of nodes of type y which match to preexisting nodes of type x, when considered over all arrivals of agents of type y.

$$\begin{aligned}
\textbf{LP-UB:}\quad & \text{maximize} && \sum_{x,y\in X} v_{xy}\alpha_{xy}\lambda_y && \\
& \text{subject to} && \alpha_{xy} \le \frac{\lambda_x}{\mu_x} && \forall x,y \in X \quad (1)\\
& && \sum_{y\in X}\alpha_{xy}\lambda_y + \sum_{y\in X}\alpha_{yx}\lambda_x \le \lambda_x && \forall x \in X \quad (2)\\
& && \alpha_{xy} \in [0,1], && \forall x,y \in X \quad (3)
\end{aligned}$$

[6] Taking the limit as the time horizon grows allows us to ignore lower-order terms.

Constraint (1) bounds the fraction of the time that some node of type y matches to some previously arrived node of type x by the probability that a node of type x is present in the system at any given time. Constraint (2) bounds the total rate at which a type can match by the total rate at which the type arrives. On the left-hand-side, the first sum captures the rate at which a type matches to those arriving after it; the second sum captures the rate at which a type matches to those who arrived before it. Note that constraints (2) and (3) together imply that $\sum_{x \in X} \alpha_{xy} \leq 1$ for all y. This makes intuitive sense: the total fraction of the time that a node matches to any preexisting type cannot be greater than 1.

We will first demonstrate that the value of LP-UB represents an upper bound on the expected value of the max-weight offline matching. Then in Section 4 we will provide a policy that garners a constant approximation of the LP value in expectation, and thus of the max-weight offline expectation.

Lemma 1. *Let v^* be the optimal value of LP-UB. Then the value of any matching policy, including policies with hindsight, is at most v^*.*

The proof of Lemma 1 is omitted due to space constraints and appears in the full version of the paper. The idea of the proof is to consider the set of agents who arrive up to some time T, and interpret the constraints of LP-UB as conditions on matchings in the induced graph of potential matches. These finite conditions include lower order terms, but these disappear when taking the limit as T grows large.

4 Online Matching Policy

We now present our online matching policy, OnlineMatch. Our policy first solves LP-UB in advance of any arrivals, and then uses the solution to guide its matching decisions. As demonstrated in the previous section, the solution to the LP-UB should be thought of as describing the optimal matching rates between types, subject to constraints that hold as time approaches infinity. Our goal is to create a policy that approximately matches the value of this LP, which we will achieve by obtaining a constant approximation to these matching rates.

Suppose that an agent, say agent i of type y, arrives at time t. The algorithm will then iterate through all potential types (including y) in a random order (line 3). For each considered type x, if there are any agents of type x present and unmatched in the market, the algorithm will select one of them arbitrarily and attempt to match it with agent i. With probability $\gamma \cdot \alpha_{xy} \cdot \max\left(1, \frac{\mu_x}{\lambda_x}\right)$ the match occurs, in which case the algorithm completes and awaits the next agent arrival. Otherwise, the algorithm moves on to the next type in X. If agent i is not matched after every $x \in X$ has been considered, then we leave agent i in the market unmatched and await the next arrival.

The match probability on line 5 deserves some discussion. This probability depends on the solution to LP-UB, and is the mechanism by which the algorithm

ALGORITHM 1: Algorithm OnlineMatch

require: scaling parameter $\gamma \in (0, 1]$
input : Online arrivals of agents

1 $(\alpha_{xy}) :=$ Solution to LP-UB;
2 **for** *each agent i arriving at time t, say of type $y \in X$* **do**
3   **for** *each type $x \in X$ in a uniformly random order* **do**
4   **if** *there is at least one unmatched agent j of type x in the market* **then**
5   match i and j with probability $\gamma \cdot \alpha_{xy} \cdot \max\left(1, \frac{\mu_x}{\lambda_x}\right)$
6   **end**
7 **end**

attempts to follow the matching rates proposed by the LP. One might be tempted to simply use α_{xy} as the match probability. However, when constructing an online policy we must consider the difference between unconditional match rates and matching rates conditional on agent types being present in the market. It may be that a particular type is extremely unlikely to be present to match during a given attempt. Consider a problem instance that includes a type x with arrival rate 1 and departure rate $1/\epsilon$, and a corresponding LP solution where $\alpha_{xy} = \epsilon$ for some y (note that this does not immediately violate any constraints, as the upper bound on α_{xy} could be as high as ϵ). The probability that any agent of type x will be present when an agent of type y arrives is at most ϵ (see Lemma 2). Thus an online policy that attempts to match agents of type x to agents of type y with probability ϵ will actually generate such a match with probability no greater than ϵ^2. In order to actually achieve the ϵ fraction that we desire, we must scale α_{xy} by $1/\epsilon$, or $\frac{\mu_x}{\lambda_x}$. Intuitively, we have scaled up the match probability according to the probability that x is present, in order to achieve the rate recommended by LP-UB. This motivates our choice of scaling factor on line 5.

The algorithm actually scales the probability by an additional factor of γ, which is a tunable parameter of the algorithm; this is to ensure that each agent has a constant probability of being available in the system unmatched when its ideal match arrives. We will optimize γ as part of our analysis of the algorithm.

Theorem 4. *Algorithm 1 is an 8-approximation to the value of LP-UB.*

4.1 Analysis

One challenge in the analysis of OnlineMatch is correlations across time: whether a certain type of agent is available in the market to be matched at time t depends on the types of other agents present in the market, as this influences matching probability. Thus, the availability of different types of agents are correlated through the pool of agents waiting to be matched at any given time. This correlation complicates the intuition that OnlineMatch will approximately mirror the aggregate match probabilities from LP-UB at every moment in time.

We address this difficulty by showing that while the evolution of which agent types are available in the market is dependent on the overall market state and correlated across types, they can be coupled with independent Poisson processes that are related via first-order stochastic dominance. That is, while agents in the market are matched at rates that vary over time with the composition of available agents, these rates are subject to uniform upper and lower bounds that reflect maximum and minimum possible matching rates. By relating to these extreme matching scenarios, we can derive uniform bounds on the success rate of matching attempts under arbitrary market conditions.

We begin by introducing the notion of an agent being *present* in the market, and bounding the probability that a node of a given type is present at any given time. We will say an agent i is present at time t if it has arrived but not yet departed; that is, if $a_i \leq t < d_i$. We'll say the node is *available* at time t if it is present and has not yet been matched to another node.

Importantly, an agent can be present but not available: even after an agent has been matched, one could simulate the departure process for that agent as though they had not matched, and we view the agent as being present until they leave under that simulated process. The advantage of considering presence, rather than availability, is that whether an agent is present at a given time depends only on their arrival and departure times, and is independent of all other agents in the market.

Lemma 2. *Choose a type $x \in X$ and any time $t \geq 0$. Then over all randomness in arrivals and departures, the probability that at least one agent of type x is present at time t is at most $\min\{\lambda_x/\mu_x, 1\}$.*

Proof. Choose some interval of time of length T, and consider all agents of type x that arrive during interval T. In expectation $\lambda_x T$ agents arrive, and each stays for an expected length of $1/\mu_x$, independently. The sum of times in market for all such agents is therefore $\lambda_x T/\mu_x$. By a union bound, the total fraction of time during which such an agent is present in the market is at most $\frac{1}{T}(\lambda_x T/\mu_x) = \lambda_x/\mu_x$. As this fraction is also at most 1, we have that the probability that such an agent is present at a given time is at most $\min(1, \lambda_x/\mu_x)$ as required.

We now wish to bound the probability that agents of a given type are present in the market, but not available. A present agent becomes unavailable in two ways: either they were matched immediately upon arriving to the market, or they are matched to another agent who arrives later. We begin by bounding the probability of the former, by showing that the occurrences of agents arriving and not being immediately matched stochastically dominate a Poisson arrival process with a reduced arrival rate.[7]

Lemma 3. *In an execution of algorithm* OnlineMatch*, consider the set of events that a node of type x arrives and is not immediately matched. The occurrence of such events stochastically dominates a Poisson arrival process of rate $\lambda_x(1 - \gamma \sum_{y \in X} \alpha_{yx})$.*

[7] Recall that if Z and Z' are two (random) point processes, then Z stochastically dominates Z' if there is a coupling of Z and Z' such that $\Pr[Z' \subseteq Z] = 1$.

Proof. Agents of type x arrive at rate λ_x. Suppose agent i of type x arrives at time t. By Lemma 2, for each $y \in X$ a node of type y is present at time t with probability at most $\min\{\lambda_y/\mu_y, 1\}$. Thus, given that Algorithm OnlineMatch considers a match with type y, this match will be successful with probability at least

$$\gamma\alpha_{yx} \max\{1, \mu_y/\lambda_y\} \cdot \min\{\lambda_y/\mu_y, 1\} = \gamma\alpha_{yx}.$$

The total probability that agent i matches to any other agent at time t is therefore at most

$$\gamma \sum_{y\in X} \alpha_{yx}$$

and hence the probability that agent i is not immediately matched is at least

$$1 - \gamma \sum_{y\in X} \alpha_{yx}.$$

We have argued that the event that a node of type x arrives and is not immediately matched is determined by an arrival process of rate λ_x, followed by a (state-dependent) random event of probability at least $1 - \gamma \sum_{y\in X} \alpha_{yx}$. This stochastically dominates an alternative event that simply takes this probability to be exactly $1 - \gamma \sum_{y\in X} \alpha_{yx}$ in all cases. But, by Fact 3, this latter process is equivalent to a Poisson arrival process of rate $\lambda_x(1 - \gamma \sum_{y\in X} \alpha_{yx})$, as required

We next consider the occurrence of events in which an agent that is currently available[8] in the market is matched to some other agent who arrives. We again connect this with a Poisson arrival process with a reduced rate.

Lemma 4. *In an execution of algorithm* OnlineMatch*, consider the event that an agent of any type arrives to the market and would match to an agent of type x if any such agent is available. The occurrence of such events is stochastically dominated by a Poisson arrival process of rate $\gamma \sum_{y\in X} \lambda_y\alpha_{xy} \max(1, \mu_x/\lambda_x)$.*

Proof. Suppose that an agent of type x is present in the market. Agents of type y arrive at rate λ_y. Consider an agent i of type y that arrives at time t. The probability that agent i matches to a node of type x is dependent on which other agents are available in the market, but is maximized when no other agents of other types are available. In the event that no other types are available, agent i will certainly consider matching to type x, in which case the match occurs with probability $\gamma\alpha_{xy} \max(1, \mu_x/\lambda_x)$.

We have argued that a node of type y arrives at rate λ_y, and then matches to an agent of type x with a state-dependent probability that is at most $\gamma\alpha_{xy} \max(1, \mu_x/\lambda_x)$. This process is stochastically dominated by one in which the match occurs with probability exactly $\gamma\alpha_{xy} \max(1, \mu_x/\lambda_x)$ upon each arrival. But, by Fact 3, this is equivalent to a Poisson arrival process of rate $\lambda_y\gamma\alpha_{xy} \max(1, \mu_x/\lambda_x)$. Summing over all types $y \in X$ completes the proof.

[8] Recall by the definition of available, such an agent is also present.

Having related availability events to independent Poisson processes, we are now ready to bound the match probabilities of OnlineMatch.

Lemma 5. *Choose any $x, y \in X$, and suppose that an agent i of type $y \in X$ arrives at time t. Then* OnlineMatch *will match i to a node of type x at time t with probability at least $\gamma(1-\gamma/2)\frac{1-\gamma}{2-\gamma}\alpha_{xy}$, where the probability is over any randomness in the algorithm and in the arrivals and departures of all other agents.*

Proof. Fix some $x, y \in X$. Suppose an agent i of type y arrives at time t, and consider the evaluation of OnlineMatch on this agent i. Some terminology: we'll say that agent i *considers* matching to an agent of type x if we enter an iteration of the loop on line 3 with type x chosen. We'll say that the agent *attempts* to match to an agent of type x if, in addition, the probabilistic match on line 5 *would* occur (regardless of whether or not the condition on line 4 evaluates to true). In other word, we can imagine pre-evaluating the probabilistic check on line 5 before checking the condition on line 4, and an attempted match corresponds to iterations in which the probabilistic check passes.

We will first bound the probability that agent i considers matching to an agent of type x. By Lemma 2, for each $z \in X$ a node of type z is present at time t with probability at most $\min\{\lambda_z/\mu_z, 1\}$. Thus, given that our algorithm considers a match with type z, this match will be successful with probability at most

$$\gamma\alpha_{zy} \max\{1, \mu_z/\lambda_z\} \cdot \min\{\lambda_z/\mu_z, 1\} = \gamma\alpha_{zy}.$$

The total probability that agent i matches to any other agent at time t is therefore at most

$$\sum_{z \in X} \gamma\alpha_{zy} \leq \gamma$$

where we used (2) from LP-UB. If we consider only half of the types $z \in X$ uniformly at random, the probability of a match is then at most $\gamma/2$ (where the expectation is over randomness in algorithm and over which types are chosen). This is a bound on the probability that the algorithm matches to some other type before type x is considered. So a match to type x will be considered by the algorithm with probability at least $(1-\gamma/2)$.

Assuming it is considered, the match will be attempted with probability $\gamma \cdot \alpha_{xy} \cdot \max(1, \mu_x/\lambda_x)$. Note that the conditional attempt probability is independent of whether the match is considered. Thus the unconditional probability that the match is attempted is at least

$$\gamma(1-\gamma/2)\alpha_{xy} \cdot \max(1, \mu_x/\lambda_x).$$

We now want to bound the probability that the attempted match to an agent of type x is successful, given that one was attempted. Consider three different events, which will occur repeatedly over time:

- Event E_1: An agent of type x arrives and is not immediately matched.

– Event E_2: An agent of any type arrives and attempts to match to an agent of type x.
– Event E_3: There is exactly one agent of type x, and that agent departs.

Suppose that the agent i of type y attempts to match to an agent of type x at time t. That match will be successful if the most recent event that occurred before t, from among events of type E_1, E_2, and E_3, is an event of type E_1.

We now consider two cases, based on the relationship between μ_x and λ_x.

Case 1: $\mu_x \leq \lambda_x$**.** Then $\max\{1, \mu_x/\lambda_x\} = 1$. By Lemma 3, the occurrences of event E_1 stochastically dominate a Poisson arrival process of rate $\lambda_1 := \lambda_x(1-\gamma \sum_{y\in X} \alpha_{yx})$. By Lemma 4, the occurrences of event E_2 are stochastically dominated by a Poisson arrival process of rate $\lambda_2 := \gamma \sum_y \alpha_{xy}\lambda_y$. And finally, occurrences of event E_3 are stochastically dominated by a Poisson arrival process of rate $\lambda_3 := \mu_x$, as this is the rate of the event when there is exactly one agent of type x present in the market (and otherwise the event cannot occur). Thus, by Fact 2, the probability that the most recent event before time t was an event of type E_1 is at least

$$\frac{\lambda_1}{\lambda_1 + \lambda_2 + \lambda_3} = \frac{\lambda_x \left(1 - \gamma \sum_{y\in X} \alpha_{yx}\right)}{\gamma \sum_{y\in X} \alpha_{xy}\lambda_y + \lambda_x \left(1 - \gamma \sum_{y\in X} \alpha_{yx}\right) + \mu_x}. \tag{4}$$

Write $q = \sum_{y\in X} \alpha_{yx}$. By constraint (3) of LP-UB, we have that $\sum_{y\in X} \alpha_{xy}\lambda_y + q\lambda_x \leq \lambda_x$, which implies $\sum_{y\in X} \alpha_{xy}\lambda_y \leq \lambda_x(1-q)$, so in particular $q \in [0,1]$. We also have $\mu_x \leq \lambda_x$ by assumption for this case analysis. The probability (4) is therefore at least

$$\min_{q\in[0,1]} \frac{1-\gamma q}{\gamma(1-q) + (1-\gamma q) + 1} = \min_{q\in[0,1]} \frac{1-\gamma q}{2 - 2\gamma q + \gamma}. \tag{5}$$

The expression in (5) is weakly decreasing in q for all $\gamma \in (0,1)$, so it achieves its minimum at $q = 1$, which is $(1-\gamma)/(2-\gamma)$. Moreover, recall that this probability is a uniform bound independent of which agents are available in the market. Thus, recalling the probability that agent i attempts to match to an agent of type x, the total unconditional probability that node i successfully matches to an agent of type x is at least

$$\gamma(1-\gamma/2)\alpha_{xy} \max(1, \mu_x/\lambda_x)\frac{1-\gamma}{2-\gamma} = \gamma(1-\gamma/2)\frac{1-\gamma}{2-\gamma}\alpha_{xy}.$$

Case 2: $\mu_x > \lambda_x$**.** Then $\max\{1, \mu_x/\lambda_x\} = \mu_x/\lambda_x$. As in case 1, Lemma 3 implies that the occurrences of event E_1 stochastically dominate a Poisson arrival process of rate $\lambda_x(1-\gamma \sum_{y\in X} \alpha_{yx})$. And occurrences of event E_3 are still stochastically dominated by a Poisson arrival process of rate μ_x. By Lemma 4, the occurrences of event E_2 are stochastically dominated by a Poisson arrival process of rate $\gamma \sum_y \alpha_{xy}\lambda_y(\mu_x/\lambda_x)$. Thus the probability that the most recent event before time t was an event of type E_1 is at least

$$\frac{\lambda_x \left(1 - \gamma \sum_{y\in X} \alpha_{yx}\right)}{\gamma \sum_{y\in X} \alpha_{xy}\lambda_y(\mu_x/\lambda_x) + \lambda_x \left(1 - \gamma \sum_{y\in X} \alpha_{yx}\right) + \mu_x}. \tag{6}$$

As in case 1, write $q = \sum_{y \in X} \alpha_{xy}$, so that $\sum_{y \in X} \alpha_{xy} \lambda_y \leq \lambda_x(1-q)$. Then $\sum_{y \in X} \alpha_{xy} \lambda_y (\mu_x/\lambda_x) \leq \mu_x(1-q)$. Also, since $\lambda_x < \mu_x$ by assumption for this case analysis, $\lambda_x \left(1 - \gamma \sum_{y \in X} \alpha_{yx}\right) < \mu_x(1-\gamma q)$. The probability (6) is therefore at most

$$\min_{q \in [0,1]} \frac{\lambda_x(1-\gamma q)}{\mu_x \gamma(1-q) + \mu_x(1-\gamma q) + \mu_x} = \min_{q \in [0,1]} \frac{\lambda_x}{\mu_x} \cdot \frac{1-\gamma q}{2 - 2\gamma q + \gamma}. \tag{7}$$

As in case 1, the expression in (7) achieves its minimum when $q = 1$, which is $\frac{\lambda_x}{\mu_x} \cdot \frac{1-\gamma}{2-\gamma}$. Thus the total unconditional probability that node i successfully matches to an agent of type x is at least

$$\gamma(1-\gamma/2)\alpha_{xy} \max(1, \mu_x/\lambda_x) \frac{\lambda_x}{\mu_x} \frac{1-\gamma}{2-\gamma} = \gamma(1-\gamma/2)\frac{1-\gamma}{2-\gamma}\alpha_{xy}.$$

Optimizing over the choice of γ, we have that $\gamma(1-\gamma/2)\frac{1-\gamma}{2-\gamma}$ takes on its maximum value at $\gamma = 1/2$, in which case $\gamma(1-\gamma/2)\frac{1-\gamma}{2-\gamma} = 1/8$. Thus, by setting $\gamma = 1/2$ in OnlineMatch, Lemma 5 implies that agents of type y arrive and match to agents of type x at rate at least $\alpha_{xy}\lambda_y/8$. The total value obtained by OnlineMatch is therefore at least $\frac{1}{8}\sum_{x,y \in X} v_{xy}\alpha_{xy}\lambda_y$, which is 1/8 of the value of LP-UB. We conclude that OnlineMatch is an 8-approximation, as required.

References

1. Akbarpour, M., Li, S., Gharan, S.O.: Thickness and information in dynamic matching markets. Journal of Political Economy **128**(3), 783–815 (2020)
2. Alaei, S., Hajiaghayi, M., Liaghat, V.: Online prophet-inequality matching with applications to ad allocation. In: Proceedings of the 13th ACM Conference on Electronic Commerce. p. 1835. EC '12, Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2229012.2229018, https://doi.org/10.1145/2229012.2229018
3. Anderson, R., Ashlagi, I., Gamarnik, D., Kanoria, Y.: Efficient dynamic barter exchange. Oper. Res. **65**(6), 1446–1459 (Dec 2017). https://doi.org/10.1287/opre.2017.1644, https://doi.org/10.1287/opre.2017.1644
4. Aouad, A., Saritaç, O.: Dynamic stochastic matching under limited time. In: Proceedings of the 21st ACM Conference on Economics and Computation. p. 789790. EC 20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3391403.3399524, https://doi.org/10.1145/3391403.3399524
5. Ashlagi, I., Burq, M., Dutta, C., Jaillet, P., Saberi, A., Sholley, C.: Edge weighted online windowed matching. In: Proceedings of the 2019 ACM Conference on Economics and Computation. pp. 729–742. EC '19, ACM, New York, NY, USA (2019). https://doi.org/10.1145/3328526.3329573, http://doi.acm.org/10.1145/3328526.3329573
6. Ashlagi, I., Burq, M., Jaillet, P., Manshadi, V.: On matching and thickness in heterogeneous dynamic markets. Operations Research **67**(4), 927–949 (2019)

7. Baccara, M., Lee, S., Yariv, L.: Optimal dynamic matching. CEPR Discussion Paper No. DP12986 (2018)
8. Dickerson, J.P., Sankararaman, K.A., Srinivasan, A., Xu, P.: Assigning tasks to workers based on historical data: Online task assignment with two-sided arrivals. In: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems. pp. 318–326. International Foundation for Autonomous Agents and Multiagent Systems (2018)
9. Feldman, J., Mehta, A., Mirrokni, V., Muthukrishnan, S.: Online stochastic matching: Beating 1-1/e. In: 2009 50th Annual IEEE Symposium on Foundations of Computer Science. pp. 117–126. IEEE (2009)
10. Gravin, N., Wang, H.: Prophet inequality for bipartite matching: Merits of being simple and non adaptive. In: Proceedings of the 2019 ACM Conference on Economics and Computation. p. 93109. EC 19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3328526.3329604, https://doi.org/10.1145/3328526.3329604
11. Haeupler, B., Mirrokni, V.S., Zadimoghaddam, M.: Online stochastic weighted matching: Improved approximation algorithms. In: Chen, N., Elkind, E., Koutsoupias, E. (eds.) Internet and Network Economics. pp. 170–181. Springer Berlin Heidelberg, Berlin, Heidelberg (2011)
12. Huang, Z., Kang, N., Tang, Z.G., Wu, X., Zhang, Y., Zhu, X.: How to match when all vertices arrive online. In: Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing. pp. 17–29 (2018)
13. Huang, Z., Peng, B., Tang, Z.G., Tao, R., Wu, X., Zhang, Y.: Tight competitive ratios of classic matching algorithms in the fully online model. In: Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 2875–2886. SIAM (2019)
14. Karp, R.M., Vazirani, U.V., Vazirani, V.V.: An optimal algorithm for on-line bipartite matching. In: Proceedings of the twenty-second annual ACM symposium on Theory of computing (1990)
15. Mahdian, M., Yan, Q.: Online bipartite matching with random arrivals: An approach based on strongly factor-revealing lps. In: Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing. pp. 597–606. STOC '11, ACM, New York, NY, USA (2011). https://doi.org/10.1145/1993636.1993716, http://doi.acm.org/10.1145/1993636.1993716
16. Mehta, A., Panigrahi, D.: Online matching with stochastic rewards. In: Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science. pp. 728–737. FOCS '12, IEEE Computer Society, Washington, DC, USA (2012). https://doi.org/10.1109/FOCS.2012.65, https://doi.org/10.1109/FOCS.2012.65
17. Mehta, A., Saberi, A., Vazirani, U.V., Vazirani, V.V.: Adwords and generalized on-line matching. In: 46th Annual IEEE Symposium on Foundations of Computer Science (2005)
18. Mehta, A., Waggoner, B., Zadimoghaddam, M.: Online stochastic matching with unequal probabilities. In: Proceedings of the Twenty-sixth Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 1388–1404. SODA '15, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2015), http://dl.acm.org/citation.cfm?id=2722129.2722221
19. Truong, V.A., Wang, X.: Prophet inequality with correlated arrival probabilities, with application to two sided matchings (2019)