Streaming Stable Matchings

Nicholas Kocurek
nkocurek@andrew.cmu.edu
Carnegie Mellon University

Harrison Wiseblatt
hwisebla@andrew.cmu.edu
Carnegie Mellon University

Abstract

The Stable Matching problem is a staple in classical complexity theory and has obvious practical applications in Big Data settings. Despite this, the problem has only been explored in a few capacities in streaming models and does not have a very complete characterization in the literature. We explore different natural modelings of the problem and the hardness of streaming in them. Specifically, we prove an $\Omega(n^2 \log n)$ lower bound on space complexity of any (even randomized) streaming algorithm exactly solving the problem. We additionally look at approaches for sublinear streaming algorithms for approximate stable matchings.

1 Introduction

In this paper we explore streaming the Stable Matching problem (often seen as the Stable Marriage, Hospital/Residents [Man16], Students/University or Students/Jobs problems [BB04] [APR05]). The problem, in its most canonical form is stated as being given two groups of equal size with strict preferences of all the members of the other group and trying to find a stable matching between the two. A matching here is a bijection between groups and stable is defined as no pairs in the matching being *unstable*, which means that they are not matched with each other but would prefer to be over their current matches.

The problem is of independent theoretical intereset due to its simplicity but is also practically relevant in many Big Data settings from large healthcare systems matching residents to hospitals to receive care to matching numerous applicants to jobs and interviews. In the 60s Gale and Shapley proved not only do stable matchings always exist but they can always be found by an extremely simple polynomial-time algorithm which has since become ubiquitous in undergraduate CS curricula [GS62]. The problem also has many equivalent forms, such as having the groups be non-bipartite (sometimes called the Stable Roommates problem) or having the preference lists be partial or non-strict. For a succinct look into some of the classical results on the Stable Matching problem and its variants we will refer the reader to [BCF08] [IM08]

Previous Work on Stable Matchings in the Streaming Setting. More recently, the problem has received interest in streaming/online settings in Big Data.

- Early work explored a certain online version of the problem where one group's preferences are revealed at the start and the other group is revealed individual by indivdual. They showed it is difficult to minimize unstable pairs in the worst case for deterministic and randomized algorithms [KMV94].
- More recent work considers an evolving model where the preference lists change over time (via swaps) but these updates are not given to the algorithm but rather must be queried. An algorithm is given that maintains a matching with $O(\log^2 n)$ unstable pairs [KLM16].
- Very recently a line of work in "reoptimization" was resolved for the Stable Matching problem. In this setting an initial instance is given and solved, but then the instance is slightly modified in a second stage and the goal is to output a new stable matching by changing the original as little as possible. It was shown that there is an optimal online algorithm in the two-stage setting and that none exists across more stages [BEY23].
- Another interesting model is the "dynamic" model which can be seen as a more natural form of the evolving model above, where preference lists can be updated and the updates are received directly by the algorithm. Stable matching in the dynamic model is more or less trivialized by considering the instance as partial and resolving with new data. Some lines of work have explored how to find "robust" matchings that are perturbed infrequently upon updating [Dov21] [AI22] [AIS23].

The Semi-streaming Model and Maximum Matchings Despite the body of work done in the various settings above, to the best of our knowledge there are no works directly addressing the problem of finding approximate/true stable matchings while minimizing the space used by the algorithm, which is a fundamental question asked in many streaming settings. The related problem of maximum bipartite matching is well explored in the so-called "semi-streaming" model where the space of the algorithm is less than the input. In this model the algorithm is given potentially multiple passes over the stream, so common approaches are to find an approximate matching and then use augmenting paths to optimize it. More novel approaches phrase the maximum weight matching problem as a linear program and use the "multiplicative weights update method" to solve the problem [KN21] [Fei+05] [KN] [Ass23] [AG11]. With the amount of literature on the maximum matching problem we find the lack of interest in the similar Stable Matching problem intriguing and attempt to provide an equivalent level of characterization.

1.1 Previous Work on Stable Matchings in Streaming or Communication

In [KMV94], it was demonstrated that any deterministic "Y-arrival" online setting has a worst-case $\Omega(n^2)$ unstable pairs, using simple counterexamples. They also prove that the "first-come-first-serve" algorithm achieves $O(n \log n)$ unstable pairs on average, which is competitive across randomized algorithms. Notably, their algorithm requires saving all of the preference lists seen so far to make the decision for the next arrival, so is not low space.

A more recent work explores the problem in a 2-way communication setting and arrives at a $\Omega(n^2)$ randomized communication complexity lower bound for Stable Matchings as well as several different relaxations to "approximate" stability. Specifically, the $\Omega(n^2)$ lower bound applies to the following problems;

1. Given an instance of the Stable Matching problem, find a stable matching.
2. Given an instance of the Stable Matching problem and a pair of participants, determine if they can be paired in some/any stable matching.
3. Given an instance of the Stable Matching problem and $0 < \varepsilon \leq 1$, find any εn many pairs that appear in some/any stable matching.

Each of these statements would be tight with the known upper bound of $O(n^2 \log n)$ which is just the complexity of sending over all the preferences of one side to the other and running Gale-Shapley.

Theorem (Informally Theorem 3.1 from [Gon+18])**.** *Finding a stable matching in the* 2*-way communication model (even with randomness) requires* $\Omega(n^2)$ *bits of communication.*

Rouhgly, the proof proceeds by a reduction from UNIQUE-DISJOINTNESS (where Alice and Bob are given subsets of $[m]$ which are either disjoint or share a single common element and need to determine which of those cases they're in). In particular, Alice and Bob are both given subsets of size $[n(n-1)]$, which we naturally view as subsets of $[n]^2 \setminus \{(i,i) : i \in [n]\}$. The (i,j) element in this matrix gives information about whether element i of each side prefers element j of the other side for each $j \neq i$, hence the size $n-1$.

Given subsets A and B, we construct the preference lists for sets X and Y as follows:

For each x_i, x_i prefers all y_j with $(i,j) \in A$ in an arbitrary order over y_i, and doesn't rank any of the other y_j's. And, for each y_j, y_j prefers all x_i with $(i,j) \in B$ in an arbitrary order over x_j, and doesn't rank any of the other x_i's. As is typical, participants prefer being matched to anyone on their preference list over being unmatched, and prefer being unmatched over being paired with anyone not on their preference list.

Then, one can verify the following claim: the matching (x_i, y_i) is the unique stable matching if and only if . The proof of uniqueness is simple but appeals to some other known results specific to stable matchings, but the proof that it is a stable matching is quite simple:

$$\begin{aligned}&(x_i, y_i) \text{ is an unstable pair}\\ \iff\ &\exists j \text{ s.t.} x_i \text{ prefers } y_j \text{ over } x_i \text{ and } y_i \text{ prefers } x_j \text{ over } y_i\\ \iff\ &(i,j) \in A \wedge (i,j) \in B.\end{aligned}$$

From this, the result of (1) follows.
The proofs of statements (2) and (3) follow similar reductions.

Although not stated in the paper, it can be seen that this immediately implies an $\Omega(n^2)$ lower bound on the space complexity of any streaming algorithm solving Stable Matchings, even under relaxations discussed in the paper. This can be done through the usual communication to streaming reduction, in this case the problems being exactly the same.

In [Gon+18], they leave open the problem of whether one can find a stable matching with εn^2 many unstable pairs in $o(n^2)$ bits of communication. There have been a few prior works providing algorithms to do this sort of approximation with other objectives in mind. We first summarize those results, and then our attempts to create an algorithm for this specific problem.

In [KP10], the authors work in the fully distributed model and define a notion of ε-approximately-stable matchings as having few pairs of participants who rank each other a ε-fraction higher on each other's preference list than their current partners. They then prove an $\Omega(\frac{\sqrt{n}}{\log n})$ lower bound for the number of communication rounds in any protocol that achieves this sort of approximation. Since ε can be arbitrarily close to 1, this feels like a very weak notion of approximation, yet the authors of [OR15] were still able to give a protocol for our notion of ε-stability that achieves $O(\log^2(n))$ communication complexity, demonstrating a separation in the difficulty of getting these different sorts of approximations. The algorithm they propose to find these almost stable matchings, appropriately called **ASM**, in each round, individuals from side X propose to batches of individuals on side Y, and side Y will temporarily accept proposals from their highest quantile, and a maximal matching is computed on the accepted proposals to determine who is matched to whom. This process is then repeated for enough iterations to guarantee convergence of a matching with $\leq \varepsilon n^2$ unstable pairs.

A similar sort of approximation is seen in [Flo+09]. They proved that, if the maximum number of individuals ranked by anyone is at most Δ, then Gale-Shapley can converge to a matching with at most εn^2 many unstable pairs in $O(\frac{\Delta^2}{\varepsilon})$ many steps. The result uses a potential function argument to show that a practically unmodified Gale-Shapley algorithm can almost converge very quickly, even if it will take many more steps to converge to a stable matching.

Two works are able to show $\Omega(n^2 \log n)$ lower bounds in certain settings, which is optimal up to constant factors since generally the entire instance takes $\Omega(n^2 \log n)$ space and Gale-Shapley can be performed in this space bound as well. The first proves that any algorithm that must query the input bit by bit must read $\Omega(n^2 \log n)$ bits to compute a stable matching [NH90]. The second considers a distributed model where all $2n$ individuals hold their own preference list and must send it to a server for a stable matching to be computed. It is shown that the server must receive

$\Omega(n^2 \log n)$, but keyly uses a technique of forcing any "holes" in the instance to become necessary to compute the stable matching, a paradigm we will mirror [CL10].

1.2 Our Results

1.2.1 Lower Bounds against Streaming Algorithms for Stable Matchings

We characterize Stable Matchings in the stream in what we believe to be the two most natural models and prove a $\Omega(n^2 \log n)$ tight-up-to-constant-factors lower bound.

Theorem (Informally Theorem 3.4)**.** *Finding a stable matching in the stream even with randomness requires $\Omega(n^2 \log n)$ space.*

The proof is by reduction from a (to our knowledge) novel communication problem inspired from Augmented INDEX. Roughly, we can think of Alice as having an encoding of n permutations on $[n]$ and Bob being interested in a single bit of Alice's, which corresponds to a single bit in one element of some permutation. However, the problem will be 1-way such that Bob cannot just ask for the bit. Intuitively then, Alice has to send most of their input to succeed with, say, constant probability. We can then reduce this problem to Stable Matchings as follows. Alice will interpret their permutations as the initial segment of half of n people and give this to a stream, sending the state over to Bob. Bob then finds out the desired bit by matching the person corresponding to the bit to the individual in the corresponding spot in their preference list. To force this, Bob will assign the other half of the people s.t. they "steal" the first few entries of the desired individual, forcing the right matching as a result.

1.3 Roadmap

Section 2 will formally define the Stable Matching problem and some common variants as well as some of the different streaming models we have identified for the problem, followed by a brief introduction to the previous work, results, and relevant techniques. In Section 3, we will prove our main result of a tight space lower bound against randomized Stable Matchings streaming algorithms. In Section 4, we will look at a relaxation to the problem that might escape both previous and our lower bounds and allow for sublinear streaming algorithms. Section 5 concludes with some future directions and acknowledgements.

2 Preliminaries

2.1 Matching Definitions

Here we will define the canonical instance that we will refer to as "The Stable Matching Problem". The problem instance is of the form $\langle X, Y, C_X, C_Y \rangle$ where X and Y are disjoint sets (sometimes referred to as vertices) with $|X| = |Y|$ and C_X and C_Y are collections of *preference lists* for X and Y respectively, where every element $x \in X$ and $y \in Y$ has a list the collection which strictly ranks all the elements of the other set. We will informally say that $x \in X$ "prefers" $y \in Y$ to $y' \in Y$ if y comes before y' in the preference list associated with x.

A matching in a stable matching instance is a bijection $\varphi : X \to Y$. We say a pair $(x, y) \in X \times Y$ is matched iff $\varphi(x) = y$. A pair $(x, y) \in X \times Y$ is *unstable* w.r.t. a matching φ if (x, y) is unmatched and x prefers y to $\phi(x)$ and y prefers x to $\phi^{-1}(y)$. A stable matching is a matching in which no pairs in $X \times Y$ are unstable.

The Stable Matching Problem can then be stated as follows.

Definition (The Stable Matching Problem)**.** *Given an instance* $\langle X, Y, C_X, C_Y \rangle$ *find and output a stable matching.*

The encoding used in the problem can vary, but we will often impose an arbitrary order on X and Y and think of them as distinct copies of $[n]$ and think of C_X and C_Y as 2D arrays where the first index determine which element's list we are specifying and the second indicating the ith most preferred element in their list. For example $C_X[i][j]$ would refer to the ith element of X's jth preferred element in Y.

In our algorithms, we will often try to approximate stable matchings, which we refer to as ε-unstable matchings. While there is no agreed-upon definition of how to define stable matchings in the literature, the definition that we will be using (unless otherwise specified) is as follows:

Definition. *Given an instance of the stable matching problem and* $0 \leq \varepsilon \leq 1$*, a* ε*-unstable matching is a matching with at most* $\varepsilon|E|$ *many unstable pairs, where* $|E| = \sum_{x \in X} |C_X[x]| + \sum_{y \in Y} |C_Y[y]|$.

In the above, we see $|E|$ is the number of edges in the directed bipartite graph with an edge $(a, b) \iff b$ is on a's preference list. For example, in the full preference list model, $|E| = n^2$ and $\varepsilon = 0$ represents a stable matching.

There are also other commonly studied variants of the model basic question, including those that allow preference lists with ties, matchings in which $|X| \neq |Y|$, etc.,. In many cases, results for these settings follow naturally from results in the setting presented above.

2.2 Streaming Models

The previous work outlined above lay out interesting streaming/online input settings that we will describe below.

- Y**-arrival Model**. In the Y-arrival model of [KMV94], we we know the entire set X and their preference lists ahead of time, and elements in Y arrive one by one along with their entire preference list. The algorithm must match each y as it arrives.

While this and other previous models provide great inspiration, none of them were designed with streaming in mind. As such, we will formally define the following new models that we think most naturally capture the Stable Matching problem:

- **Full Preference List Model**. In this model the algorithm receives in the stream a preference list of one individual at a time, much like with the Y vertices in the Y-arrival model. The difference in this model is that the algorithm also receives X's preference lists in the stream instead of knowing them ahead of time. No ordering is imposed on which vertices arrive when. The preference list is encoded as a list of n numbers, so takes $\Theta(n \log n)$ space. Since there are $2n$ vertices, the total input size is $\Theta(n^2 \log n)$.

- **Partial Preference List Model**. This model is a relaxation of the Full Preference List Model above, where the algorithm receives individual entries to preference lists. Once again ordering can be arbitrary, in particular you do not have to receive all of one preference list to start receiving another. The items in the stream can be encoded as (x, y, i), indicating that the ith element on x's preference list is y. Each tuple then takes $\Theta(\log n)$ space and there $2n^2$ many, totalling to $\Theta(n^2 \log n)$ space again.

3 Lower Bounds for Streaming Stable Matchings

In this section we formally prove Theorem 3.4. We will do this in three stages. First we will show an intuitive way to trick low space deterministic algorithms. We will then adapt this technique in a more classic communication complexity reduction to achieve an $\Omega(n^2)$ lower bound for the randomized case. We will then show how to augment our communication problem to get a more robust lower bound.

3.1 A First Attempt: An $\Omega(n^2 \log n)$ Lower Bound for Deterministic Algorithms

Theorem 3.1. *Any deterministic streaming algorithm in the Partial Preference List Model solving the Stable Matching Problem with error probability has space complexity* $\geq \Omega(n^2 \log n)$.

Proof:
We will construct $\Theta(n^2 \log n)$ Stable Matchings inputs for any fixed deterministic streaming algorithm. By PHP, any streaming algorithm using $o(n^2 \log n)$ space will necessarily end up in the same state receiving some pair of these inputs. We will then append the rest of the stream in a way that our original two inputs end up with different answers, yet they are indistinguishable to the algorithm.

Formally, the set of inputs will just specify that for $i \in [n/2]$ in X, we will assign the first half of $C_X[i]$, the ith elements preference list, to be some permutation on $[n/2]$. It is a simple counting argument that there are $((n/2)!)^{n/2}$, which by Stirling's approximation requires $\log_2(((n/2)!)^{n/2}) = \Omega(n^2 \log n)$ bits to encode.

Assume then that we get two inputs I_1 and I_2 that end up in the same state when fed to a fixed algorithm in the stream. Since $I_1 \neq I_2$, there must be some element that differs. Let $x \in X$ be the first element whose preference list is different between the two, and let $j \in [n/2]$ be the first index in $C_X[x]$ that differs between the two. The adversarial stream we will construct is as follows.

- For $i \in [j-1]$, $C_X[n/2+i][1] = C_X[x][i]$
- For $y \in Y, i \in [j-1]$, $C_Y[y][i] = n/2 + i$
- For $y \in Y$, $C_Y[y][j] = x$
- Everything else can be chosen arbitrarily (but should be consistent with above)

To see that this works it is helpful to have the following claim.

Claim. Any Stable Matchings instance where one set has every element the same preference list has a unique stable matching.

Proof:
WLOG let all of Y's preference lists be $[n]$. Clearly any matching where 1 is not matched to their

most preferred partner is unstable, so match them. The stability of the rest of the matching then depends on the stability of the subinstance where 1 is removed from the front of all Y's lists and 1's partner is removed from all of X's list. All of Y still has the same preference list however, so we just proceed inductively.

Using this claim, it is not hard to see what the instance above is doing: we are just forcing x to be matched to its jth preferred individual, but since this differs in I_1 and I_2 they won't have the same stable matching.

More formally, we can imagine running Gale-Shapley with members of X proposing in order $n/2+1, n/2+2, ..., n/2+j-1, x....$ Clearly $n/2+i$ will take x's ith preferred individual, so when x proposes they will get their jth most preferred. No unmatchings happen as we are proposing in the order Y most prefers. Since their is only one stable matching, this corresponds to it. We conclude then that any deterministic algorithm cannot always be correct using $o(n^2 \log n)$ space.

3.2 A Second Attempt: An $\Omega(n^2)$ Lower Bound for Randomized Algorithms

In this section we attempt to extend to the previous result to cover randomized algorithms by using mostly the same technique inside of a reduction from a communication problem. It was originally suggested to us by Professor Woodruff to use INDEX. For some subtle technical reasons this doesn't work, but we can easily modify INDEX to make it work. Our modification creates the following problem.

Definition (MULTI-PERM Problem)**.** *Let Alice have n permutations on $[n]$, each encoded as S, a list of n numbers in permutation order. Assume each number is encoded in $\log_2 n$ bits. Assume Bob has $i \in [n^2 \log n]$ and wants to recover S_i. Communication goes from Alice to Bob. A protocol is said to solve the problem if for any S and i Bob can output S_i with error probability ε over the choices of Alice and Bob.*

This problem is difficult to communicate.

Claim. Any protocol solving MULTI-PERM with error probability $\varepsilon < \frac{1}{2}$ requires $\Omega(n^2 \log n)$ communication.

We will defer the proof of the claim until Section 3.3. We are now ready to prove our first lower bound for randomized streaming algorithms for Stable Matchings.

Theorem 3.2. *Any randomized streaming algorithm in the Full Preference List Model solving the Stable Matching Problem with error probability $\varepsilon < \frac{1}{2}$ has space complexity $\geq \Omega(n^2)$.*

Note that by the fact that the Full Preference List Model is a restriction to the Partial Preference List Model, we immediately obtain the following corollary.

Corollary 3.3. *The same space lower bounds hold for the Partial Preference List Model.*

Proof of Theorem 3.2:
Assume that GS is a streaming algorithm in the Full Preference List Model that solves the stable matching problem with error probability $\varepsilon \geq 0$ and space complexity $S(n)$. Let Alice hold an encoding of $\frac{n}{2}$ permutations of $[n/2]$ as in the MULTI-PERM problem. Letting $X = [n]$ and $Y = [n]$,

Alice simply sets the preference list C_X for each $i \in [n/2]$ as $C_X[i] =$ the ith encoded permutation in s. Since this is the full preference setting, Alice also needs to set $C_X[i]$ for $i \in [n] \setminus [n/2]$ simultaneously, which can just be arbitrary (but consistent with the encoded permutation). Alice now runs GS on the above input (order can be arbitrary) up to Δ times independently, and sends the state of all Δ runs as a list L.

Upon receiving this and an index $i \in [n^2 \log n]$, Bob computes which which index it corresponds to in the 2D array C_X, identifying some $x \in X$ and some jth index of the x's preference list. It suffices to recover $C_X[x][j]$ to determine s_i. Bob then runs the following algorithm.

```
def Algorithm 1$(L, x, j)$ :
1.  Set remainder $C_X$ preference lists arbitrarily
2.  Set $C_Y$ preference lists s.t. $C_Y[y][1] = x$ then the rest arbitrarily
3.  for i = 1 to j:
4.      $M =$ boostedGS($\langle C_X, C_Y\rangle, L$)
5.      $C_X[n/2+i][1] =$ $x$'s matched partner in $M$.
6.      for y in Y:
7.        $C_Y[y][i] = n/2 + i$
8.        $C_Y[y][i+1] = x$
9.        Do the above two as swaps (some element then gets moved down the list)
10. return $x$'s matched partner in $M$
```

Here "boostedGS" is a subroutine that takes the remaining lists in C_X and C_Y and feeds them (in any order) in stream into the Δ runs given in L, then takes a majority over the outputted matchings.

Claim. Algorithm 1 recovers s_i with probability $\frac{99}{100}$ given $\Delta = \Omega(\log n)$.

Proof:
We will delay all probabilistic analysis by first conditioning on the following set of events. Let E_i be the event that "boostedGS" returns a stable matching on the ith iteration.

$$E = \{[E_k \mid E_{<k}] \mid k \in [j], \text{ where } E_{<k} \text{ is all } E_i \text{ before } k\}$$

Essentially this is conditioning that every run of "boostedGS" is correct given the previous runs have all been correct. It is not hard to argue correctness inductively from here: namely that x is matched to its ith most preferred partner on iteration i.

On the first iteration, $C_Y[y][1] = x$ for all $y \in Y$, so clearly in any stable matching outputted x gets matched to $C_X[x][1]$, so x's matched partner is its most preferred match. Now inductively assume that on iteration k, x was matched with its ith most preferred partner on all previous iterations. Necessarily then for all $y \in Y$, $C_Y[y][i] = n/2 + i$ for all $i \in [k-1]$ and $C_Y[y][k] = x$. Moreover, $C_X[n/2+i][1]$ is x's ith most preferred partner for $i \in [k-1]$. So when we compute the new stable matching, which is correct by our conditioning, the following happens. Clearly each $n/2 + i \in X$, gets to choose their partner in order $i \in [k-1]$ as they are the most preferred by all $y \in Y$. Each will choose the ith preferred partner of x, since they are their most preferred and have not been taken previously. This leaves x to choose with all of its first $k-1$ preferred partners taken, but its k most preferred partner has not been taken, matching them. By this inductive argument then, after the jth iteration x will be matched to $C_X[x][j]$, from which we can recover the goal bit in s.

Now the success probability comes directly from computing the probability of event E.

Claim. Event E occurs with probability $\frac{99}{100}$ given $\Delta = \Omega(\log n)$.

Proof:
The proof is a simple boosting + Chernoff bound + union bound argument. Note that the event $[E_i \mid E_{<i}]$ is just that a majority of independent runs of GS are correct on a *specific instance*. Indeed, the instance is fixed before the algorithm even runs and thus the probability is just the probability that the streaming algorithm is correct on this specific instance. So to compute the probability E does not occur, it suffices to union bound over the algorithm being incorrect on j separate instances. Letting $\mathcal{E}$ be the probability that "boostedGS" fails on a specific instance gives us the followinng.

$$\Pr[\bar{E}] \leq \sum_{i \in 1}^{j} \mathcal{E} = j \cdot \mathcal{E}$$

To derive $\mathcal{E}$, let F_i for $i \in [\Delta]$ be an indicator for event that the ith run of GS on some arbitrary input is incorrect and $F = \sum_{i=1}^{\Delta} F_i$. As the runs are independent, the variables are as well. By linearity of expectation we easily get that $\mathbb{E}[F] = \varepsilon\Delta$. We are then interested in bounding the probability that $F \geq \Delta/2$, as otherwise the algorithm is correct on over half the calls and we deduce the majority matching is stable. There is a technicality: we want the stable matching to be unique so it truly is majority but this is the case per Section 3.1. So we proceed via Hoeffding's inequality.

$$\Pr[F \geq \Delta/2] = \Pr[F \geq \mathbb{E}[F] + (1/2 - \varepsilon)\Delta] < e^{-2(1/2-\varepsilon)^2\Delta}$$

It follows that this is less than $\frac{1}{100n}$ when $\Delta \geq \frac{1}{2(1/2-\varepsilon)^2}\ln(100n)$, so $j \cdot \mathcal{E} \leq \frac{1}{100}$ as desired.

Claim. $S(n) \geq \Omega(n^2)$.

Proof:
The protocol above uses at most $\Delta \cdot S(n)$, which just accounts for the maximum space any single run of the stream can do. The lower bound from MULTI-PERM is $\Omega(n^2 \log n)$, so necessarily $\Delta \cdot S(n) \geq \Omega(n^2 \log n)$. Since we can take $\Delta = O(\log n)$, we conclude $S(n) \geq \Omega(n^2)$.

The last claim to prove is that when $\varepsilon = O(1/n)$, we can achieve $S(n) \geq \Omega(n^2 \log n)$, which we can accomplish by setting $\Delta = O(1)$.

3.3 A Final Attempt: An $\Omega(n^2 \log n)$ Lower Bound for Randomized Algorithms

Doing a post-mortem of the proof above, the largest bottleneck to achieving $\Omega(n^2 \log n)$ was that the number of runs of the streaming algorithm could be $\Omega(n)$. Indeed, it is believable that this is not a weakness of the Chernoff bound: any algorithm with constant failure probability *should* compound error across $\Omega(n)$ runs, so the obvious candidate for improvement is cutting down the number of runs.

Our game plan is as such then: how do we eliminate the number of runs needed in Algorithm 1. The information we need is what the first so-and-so many elements of our identified preference list is, which is what we recover in the runs. The solution then is to give us exactly this information: along with a target index in this preference list we will give Bob all the previous elements of the list as well. Our concern would be that this weakens the communication problem, but the key insight we will show is that this does not help in the communication setting enough for us to lose hardness.

We now move to formally prove our main result, Theorem 3.4.

Theorem 3.4. *Any randomized streaming algorithm in the Full Preference List Model solving the Stable Matching Problem with error probability $\varepsilon < \frac{1}{2}$ has space complexity $\geq \Omega(n^2 \log n)$.*

Proof:
The proof proceeds as described above. We will use the communication protocol from Section 3.2 but can skip to the last iteration instance, which is just the instance we constructed in Section 3.1, by using direct access to the more preferred elements we need. This communication protocol can be formalized as follows.

Definition (Augmented MULTI-PERM Problem)**.** *Let Alice have n permutations on $[n]$, each encoded as S, a list of n numbers in permutation order. Assume each number is encoded in $\log_2 n$ bits. Let Bob have $i \in [n^2]$ which corresponds to some jth element in some $x \in X$'s preference list. Let Bob additionally have access to the first $j-1$ elements of x's preference list. In this version, Bob should have to output the entire right element $S^{(x,j)}$.*

The theorem (as well as more or less the hardness of MULTI-PERM) follows directly from the communication complexity of Augmented MULTI-PERM.

Theorem 3.5. *Any protocol solving Augmented MULTI-PERM with error probability $\varepsilon < \frac{1}{2}$ requires $\Omega(n^2 \log n)$ communication.*

Proof:
We proceed in the usual way, showing $|M| \geq H(M) \geq I(S; M)$ (here H and I are entropy and mutual information), so it suffices to show the mutual information of S and M must be high in any protocol. Let $S^{(i)}$ be the ith permutation in S. By conditioning we have the following.

$$I(S; M) = \sum_{i=1}^{n} H(S^{(i)} \mid S^{(<i)}) - H(S^{(i)} \mid M, S^{(<i)})$$

Conveniently, the permutations are independent of one another, so $H(S^{(i)} \mid S^{(<i)}) = H(S^{(i)}) = \log_2(n!) = \Theta(n \log n)$. Since conditioning cannot raise entropy then, we can look at $H(S^{(i)} \mid M)$ as an upper bound on the second term. From here we will break up each $S^{(i)}$ across its elements by conditioning.

$$I(S; M) \geq \Omega(n^2 \log n) - \sum_{i=1}^{n} H(S^{(i)} \mid M) = \Omega(n^2 \log n) - \sum_{i=1}^{n} \sum_{j=1}^{n} H(S^{(i,j)} \mid M, S^{(i,<j)})$$

Now we will use Fano's Inequality on the Markov Chain $S \to M, S^{(i,<j)} \to S'$, which says that $H(S^{(i,j)} \mid M, S^{(i,<j)}) \leq H(\varepsilon) + \varepsilon \log_2(n-1)$ since by assumption that the MULTI-PERM protocol has $\Pr[S^{(i,j)} \neq S'^{(i,j)}] \leq \varepsilon$. Here $\log_2(n-1)$ comes from the fact that $S^{(i,j)}$ can take on n values.

From here we plug in and conclude $I(S;M) \geq \Omega(n^2 \log n)$ as the constant suppressed above is close to 1 which is greater than ε.

4 Sublinear Streams for Approximate Stable Matchings

In light of Theorem 3.5, we now have a strong guarantee against streaming algorithms that exactly solve Stable Matchings, so the natural next step is to relax the problem. For this, we will use the notion of ε-unstable matchings.

Conjecture 4.1. For any $c \in [0,1]$, there is a randomized streaming algorithm in the Full Preference List Model that finds a matching with at most $O(n^{1+2c} \log n)$ unstable pairs with space complexity $O(n^{2-c} \log n)$.

We consider and analyze the following algorithm.

```
def Algorithm 2($\langle C_X, C_Y \rangle$) :
1.  $S_X$ = Sample $T$ individuals from $X$
2.  $S_Y$ = Sample $T$ individuals from $Y$
3.  In stream, save only $S_X$ and $S_Y$'s preference lists
4.  for every matching $M$ of $X$ and $Y$
5.      $count = 0$
6.      for $(x, y)$ in $S_X \times S_Y$:
7.        if $(x, y)$ is unstable w.r.t.$M$:
8.          $count = count + 1$
9.      if $count \leq n \log n$ then return $M$
10.  return whatever
```

Claim 1. Algorithm 2 uses $O(Tn \log n)$ space.

Claim 2. Algorithm 2 returns a matching with $O(b)$ unstable pairs with error probability ε given $T^2 b \geq \Omega(n^3 \log n)$.

The two claims would be sufficient to prove Conjecture 4.1 by letting $T = \Theta(n^{1-c})$ and $b = \Theta(n^{1+2c} \log n)$. However, we will show that our analysis for Claim 2 falls through, but we include it as the sketch does not seem unable to be recovered.

Proof of Claim 1:
The main space contributor is storing the $2T$ preference lists for S_X and S_Y, which eats up $2Tn \log n$ space plus whatever overhead. To iterate through matchings we can simply store each matching as a list of n numbers, taking $O(n \log n)$ space. Every other counter can be stored in roughly $O(\log n)$.

Sketch of Claim 2:
At a high-level, we will try to show that any matching with a large number of unstable pairs is likely to be well-sampled. So likely in fact, that we will be able to union bound over all matchings. This establishes that any largely unstable matching will not appear stable in our sample, so we can

then just output the first decent matching according to our sample.

More formally, fix a matching M which has $\geq b$ unstable pairs w.r.t. our instance. It is worth noting that determining exactly how many unstable pairs a given matching has seems difficult to do in our space bound, but this is unnecessary for the following argument. It is not hard to see that each unstable pair is included in the count with probability $(T/n)^2$, just the probability that each of its individuals is sampled. In this case, the algorithm will easily see that both are higher on the other's list than its proposed partner in M. From here, if we set $T^2 b \geq \Omega(n^3 \log n)$ the expected number of unstable pairs that carry into our sample is $\geq \Omega(n \log n)$. If we could show sufficient concentration, that is $\leq \frac{1}{n^n}$ probability of dropping below $n \log n$, then we could union bound across all matchings and conclude that M will not be output.

The problem that arises is that whether or not an unstable pair appears in our sample is very much dependent on other unstable pairs appearing, so using a strong concentration bound like the Chernoff bound does not seem possible.

To rid ourselves of this dependence, one could try to sample only information about each pair separately like below. Note this algorithm assumes the Partial Preference List Model.

```
def Algorithm 3($\langle C_X, C_Y \rangle$) :
1.  $S$ = Sample each pair $(x, y)$ w.p. $p$
2.  In stream, save only info between $x$ and $y$'s in $S$
3.  for every matching $M$ of $X$ and $Y$
4.    $count = 0$
5.    for $(x, y)$ in $S$:
6.      if $(x, y)$ is unstable w.r.t.$M$:
7.        $count = count + 1$
8.    if $count \leq n \log n$ then return $M$
9.  return whatever
```

This kind of sampling indeed avoids dependence between the events of interest, but it runs into the issue that deciding whether (x, y) is unstable with respect to M is difficult without having all of x's and y's preference lists, and keeping the entire preference list for any element in any pair in S is not feasible as in general S contains every element unless p is too small to get a good sample.

Indeed, our final attempt will not try to improve on our sampling method, but instead will choose our matching more cleverly so that the event we need to avoid is much weaker.

```
def Algorithm 4($\langle C_X, C_Y \rangle$) :
1.  $S_X$ = Sample $T$ individuals from $X$
2.  $S_Y$ = Sample $T$ individuals from $Y$
3.  In stream, save only $S_X$ and $S_Y$'s preference lists
4.  for every matching $M$ of $X$ and $Y$
5.    $count = 0$
6.    for $(x, y)$ in $S_X \times S_Y$:
7.      if $(x, y)$ is unstable w.r.t.$M$:
8.        $count = count + 1$
9.    if $count = 0$ then return $M$
10. return whatever
```

Claim 3. Algorithm 4 uses $O(Tn \log n)$ space.

Proof: Follows from Claim 1 for Algorithm 2.

Claim 4. Algorithm 4 returns a matching with $O(b)$ unstable pairs with error probability ε given $T^2 b \geq \Omega(n^3 \log n)$.

Proof:
As above we will attempt to prove the contrapositive, that any M with $\geq b$ unstable pairs will not be output with high probability.

The key insight here is something that was swept under the rug in the first attempt - the completeness of the algorithm. It is a priori not apparent that this algorithm terminates in the loop: why should there be a matching with *count* $= 0$? The reason is quite simply the Gale-Shapley algorithm. Gale-Shapley guarantees on the *induced* subinstance from S_X and S_Y there is a stable matching. Individuals outside the subinstance can be matched arbitrarily but importantly none of the sampled individuals can be unstable w.r.t. such a matching, as this would contradict the stability of the subinstance.

The contrapositive becomes easier: we just need to show that for any M with $\geq b$ unstable pairs, we cannot have *count* $= 0$ with high probability. From our analysis above, we recall that *any* unstable pair being sampled will increase the count, so any matching that slips past must have that $S_X \times S_Y$ avoids *all* unstable pairs.

This pathway suggests the following approach: show that any matching with "many" unstable pairs cannot have many subsets that it is stable on (where the stability is w.r.t. the subinstance induced). Namely, in the $T = n^{1-c}$ regime, to get the $1/n^n$ failure probability we wanted above, we need the number to be something like poly(n).

5 Discussion

Here we leave a couple unresolved questions related to our results.

- In our exploration we improved the previous $\Omega(n^2)$ lower bound on randomized Stable Matchings streaming to $\Omega(n^2 \log n)$. The previous lower bound also held in some relaxed settings where the outputted must be approximately stable, not unrelated to the notion explored above. A natural question might be if algorithms cannot be found in these settings, does the $\Omega(n^2 \log n)$ bound extend. It is not immediately obvious how to extend our technique, which heavily relies on the exact pair we care about being matched correctly.

- Our biggest open problem is Conjecture 4.1 above, where we have already explored many approaches and left some open-ended. It is worth noting that this result mirrors the results of previous work quite well. The most interesting regimes are when $b = \Theta(n \log n)$ and $T = \Theta(n)$ and when $b = \Theta(n^2)$ and $T = \Theta(\sqrt{n \log n})$. The former says roughly that by sampling a constant fraction of our input, we would be able to find a matching with roughly $n \log n$ unstable pairs, improving greatly on the trivial n^2. The second would say that we can find a matching with some constant factor better than the trivial n^2 but with only $n^{3/2} \log n$ space, which is truly sublinear. The last interesting piece is that our approach inherently seems to greatly fail to get a $\Theta(n)$ unstable pairs matching, which more or less mimics the previous work of [Gon+18] that a related regime is hard to communicate.

Acknowledgements. The authors would like to thank David Woodruff and Honghao Lin for their support on this project through the 15-851: Algorithms for Big Data course at CMU. In particular, we thank Woodruff for the suggestion of converting our deterministic lower bound into a randomized one through INDEX and the sampling approach for streaming.

References

[GS62] D. Gale and L. S. Shapley. “College Admissions and the Stability of Marriage”. In: *The American Mathematical Monthly* 69.1 (1962), pp. 9–15. ISSN: 00029890, 19300972. URL: http://www.jstor.org/stable/2312726 (visited on 04/03/2024).

[NH90] Cheng Ng and Daniel S. Hirschberg. “Lower Bounds for the Stable Marriage Problem and Its Variants”. In: *SIAM Journal on Computing* 19.1 (1990), pp. 71–77. DOI: 10.1137/0219004. eprint: https://doi.org/10.1137/0219004. URL: https://doi.org/10.1137/0219004.

[KMV94] Samir Khuller, Stephen G. Mitchell, and Vijay V. Vazirani. “On-line algorithms for weighted bipartite matching and stable marriages”. In: *Theoretical Computer Science* 127.2 (1994), pp. 255–267. ISSN: 0304-3975. DOI: https://doi.org/10.1016/0304-3975(94)90042-6. URL: https://www.sciencedirect.com/science/article/pii/0304397594900426.

[BB04] Mourad Baiou and Michel Balinski. “Student admissions and faculty recruitment”. In: *Theoretical Computer Science* 322.2 (2004). Discrete Applied Problems - Florilegium for E. Goles, pp. 245–265. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2004.03.011. URL: https://www.sciencedirect.com/science/article/pii/S0304397504001628.

[APR05] Atila Abdulkadiroglu, Parag A. Pathak, and Alvin E. Roth. “The New York City High School Match”. In: *American Economic Review* 95.2 (May 2005), pp. 364–367. DOI: 10.1257/000282805774670167. URL: https://www.aeaweb.org/articles?id=10.1257/000282805774670167.

[Fei+05] Joan Feigenbaum et al. “On graph problems in a semi-streaming model”. In: *Theor. Comput. Sci.* 348.2 (Dec. 2005), pp. 207–216. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2005.09.013. URL: https://doi.org/10.1016/j.tcs.2005.09.013.

[BCF08] Péter Biró, Katarína Cechlárová, and Tamás Fleiner. “The dynamics of stable matchings and half-matchings for the stable marriage and roommates problems”. In: *International Journal of Game Theory* 36.3 (Mar. 2008), pp. 333–352. ISSN: 1432-1270. DOI: 10.1007/s00182-007-0084-3. URL: https://doi.org/10.1007/s00182-007-0084-3.

[IM08] Kazuo Iwama and Shuichi Miyazaki. “A Survey of the Stable Marriage Problem and Its Variants”. In: *International Conference on Informatics Education and Research for Knowledge-Circulating Society (icks 2008)*. 2008, pp. 131–136. DOI: 10.1109/ICKS.2008.7.

[Flo+09] Patrik Floréen et al. “Almost Stable Matchings by Truncating the Gale–Shapley Algorithm”. In: *Algorithmica* 58.1 (Aug. 2009), pp. 102–118. ISSN: 1432-0541. DOI: 10.1007/s00453-009-9353-9. URL: http://dx.doi.org/10.1007/s00453-009-9353-9.

[CL10] Jen-Hou Chou and Chi-Jen Lu. “Communication Requirements for Stable Marriages”. In: *Algorithms and Complexity*. Ed. by Tiziana Calamoneri and Josep Diaz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 371–382. ISBN: 978-3-642-13073-1.

[KP10] Andrew Kipnis and Boaz Patt-Shamir. “On the Complexity of Distributed Stable Matching with Small Messages”. In: *Distributed Computing* 23 (2010), pp. 151–161. DOI: 10.1007/s00446-010-0105-5. URL: https://link.springer.com/article/10.1007/s00446-010-0105-5.

[AG11] Kook Jin Ahn and Sudipto Guha. *Linear Programming in the Semi-streaming Model with Application to the Maximum Matching Problem.* 2011. arXiv: `1104.2315 [cs.DS]`.

[OR15] Rafail Ostrovsky and Will Rosenbaum. *Fast distributed almost stable marriages.* 2015. arXiv: `1408.2782 [cs.GT]`.

[KLM16] Varun Kanade, Nikos Leonardos, and Frédéric Magniez. "Stable Matching with Evolving Preferences". In: *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2016).* Ed. by Klaus Jansen et al. Vol. 60. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016, 36:1–36:13. ISBN: 978-3-95977-018-7. DOI: `10.4230/LIPIcs.APPROX-RANDOM.2016.36`. URL: `https://drops-dev.dagstuhl.de/entities/document/10.4230/LIPIcs.APPROX-RANDOM.2016.36`.

[Man16] David F. Manlove. "Hospitals/Residents Problem". In: *Encyclopedia of Algorithms.* Ed. by Ming-Yang Kao. New York, NY: Springer New York, 2016, pp. 926–930. ISBN: 978-1-4939-2864-4. DOI: `10.1007/978-1-4939-2864-4_180`. URL: `https://doi.org/10.1007/978-1-4939-2864-4_180`.

[Gon+18] Yannai A. Gonczarowski et al. *A Stable Marriage Requires Communication.* 2018. arXiv: `1405.7709 [cs.GT]`.

[Dov21] Laura Doval. *Dynamically Stable Matching.* 2021. arXiv: `1906.11391 [econ.TH]`.

[KN21] Christian Konrad and Kheeran K. Naidu. "On Two-Pass Streaming Algorithms for Maximum Bipartite Matching". en. In: Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2021. DOI: `10.4230/LIPICS.APPROX/RANDOM.2021.19`. URL: `https://drops.dagstuhl.de/entities/document/10.4230/LIPIcs.APPROX/RANDOM.2021.19`.

[AI22] Akhmad Alimudin and Yoshiteru Ishida. "Matching-updating mechanism: A solution for the stable marriage problem with dynamic preferences". en. In: *Entropy (Basel)* 24.2 (Feb. 2022), p. 263.

[AIS23] Akhmad Alimudin, Yoshiteru Ishida, and Koutarou Suzuki. "Maintaining Stability for a Matching Problem Under Dynamic Preference". In: *IEEE Access* 11 (2023), pp. 24203–24215. DOI: `10.1109/ACCESS.2023.3243245`.

[Ass23] Sepehr Assadi. *A Simple* $(1-\epsilon)$*-Approximation Semi-Streaming Algorithm for Maximum (Weighted) Matching.* 2023. arXiv: `2307.02968 [cs.DS]`.

[BEY23] Evripidis Bampis, Bruno Escoffier, and Paul Youssef. *Online 2-stage Stable Matching.* 2023. arXiv: `2207.02057 [cs.DS]`.

[KN] Christian Konrad and Kheeran K. Naidu. "An Unconditional Lower Bound for Two-Pass Streaming Algorithms for Maximum Matching Approximation". In: *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 2881–2899. DOI: `10.1137/1.9781611977912.102`. eprint: `https://epubs.siam.org/doi/pdf/10.1137/1.9781611977912.102`. URL: `https://epubs.siam.org/doi/abs/10.1137/1.9781611977912.102`.