Near-popular matchings in the Roommates problem*

Chien-Chung Huang[1] and Telikepalli Kavitha[2]

[1] Humboldt-Universität zu Berlin, Germany. villars@informatik.hu-berlin.de
[2] Tata Institute of Fundamental Research, India. kavitha@tcs.tifr.res.in

Abstract. Our input is a graph $G = (V, E)$ where each vertex ranks its neighbors in a strict order of preference. The problem is to compute a matching in G that captures the preferences of the vertices in a *popular* way. Matching M is more popular than matching M' if the number of vertices that prefer M to M' is more than those that prefer M' to M. The *unpopularity factor* of M measures by what factor any matching can be more popular than M. We show that G always admits a matching whose unpopularity factor is $O(\log |V|)$ and such a matching can be computed in linear time. In our problem the optimal matching would be a *least* unpopularity factor matching - we show that computing such a matching is NP-hard. In fact, for any $\epsilon > 0$, it is NP-hard to compute a matching whose unpopularity factor is at most $4/3 - \epsilon$ of the optimal.

* Supported by IMPECS (the Indo-German Max Planck Center for Computer Science). A preliminary version of this work appeared in the 19th Annual European Symposium on Algorithms (ESA) 2011, LNCS 6942, pages 167-179.

1 Introduction

Let V be a set of n people, each of whom seeks a roommate. Every person has a strict ranking over those people in V whom he/she considers as acceptable roommates. This can be modeled as a graph $G = (V, E)$ where each $u \in V$ ranks its neighbors in a strict order of preference. Preference lists can be incomplete, i.e., a vertex need not be adjacent to all the other vertices. Each vertex seeks to be matched to a neighbor and for any $u \in V$, if u ranks v higher than v' in its preference list, then u prefers v as its roommate to v'. This is called an instance of the *roommates* problem with incomplete lists, which is a generalization of the *stable marriage* problem.

In the stable marriage problem, the input graph G is bipartite and the problem is to compute a matching M in G that admits no blocking edges: such a matching M is called a *stable* matching. An edge (a, b) is a *blocking edge* to M if a is either unmatched or prefers b to $M(a)$ and similarly, b is either unmatched or prefers a to $M(b)$. While every bipartite graph admits a stable matching [3], it is easy to come up with roommates instances that admit no stable matchings. Consider the instance H on 3 vertices $\{a_0, a_1, a_2\}$ where for $i = 0, 1, 2$, a_i prefers a_{i+1} to a_{i+2} (all subscripts are modulo 3). None of the matchings $M_0 = \{(a_0, a_1)\}$, $M_1 = \{(a_1, a_2)\}$, $M_2 = \{(a_2, a_3)\}$ is stable since each M_i (for $i = 0, 1, 2$) admits a blocking edge (a_{i+1}, a_{i+2}) (the subscripts are modulo 3).

Given a roommates instance G, there is a linear time algorithm by Irving [7] to determine if G admits a stable matching or not and if so, to compute one. Our goal now is to come up with a notion of optimality that is weaker than stability such that a matching satisfying this optimality criterion always exists in G. We also want such an optimal matching to be computed efficiently.

1.1 Popular Matchings

Popularity is a weaker notion than stability. We define popular matchings below. For any two matchings M and M', we say that vertex u prefers M to M' if u is *better off* in M than in M' (i.e., u is either matched in M and unmatched in M' or matched in both and prefers $M(u)$ to $M'(u)$).

Let $\phi(M, M') =$ the number of vertices that prefer M to M'. We say that M' is more popular than M if $\phi(M', M) > \phi(M, M')$.

Definition 1. *A matching M is popular if there is no matching that is more popular than M, i.e., $\phi(M, M') \geq \phi(M', M)$ for all matchings M' in G.*

It is easy to see that every stable matching is popular [2]: while comparing a stable matching S with any matching M, observe that for any edge $(u, v) \in M \setminus S$, *both* u and v cannot prefer M to S; otherwise (u, v) would be a blocking edge to S, contradicting its stability. Also, no vertex left unmatched by M can prefer M to S. Thus summing all the votes, it follows that $\phi(M, S) \leq \phi(S, M)$. A simple instance from [2] that admits no stable matching but has popular matchings is the following: consider the graph H described above with a fourth vertex a_3 added to this instance such that a_3 is at the tail of the preference lists of a_0, a_1, and a_2. The vertex a_3's preference list is a_0 followed by a_1, and then a_2. It can be checked that this instance has no stable matching, however it admits 2 popular matchings: $\{(a_0, a_3), (a_1, a_2)\}$ and $\{(a_1, a_3), (a_0, a_2)\}$.

Since there is no matching where more vertices are better-off than in a popular matching, a popular matching is a natural candidate for an optimal matching. But popular matchings also do not always exist. The instance H on $\{a_0, a_1, a_2\}$ described earlier, is one such instance. In any instance G, let us measure by what factor one matching (say, M_1) can be more popular than another (say, M_0) as follows:

$$\Delta(M_0, M_1) = \begin{cases} \frac{\phi(M_1, M_0)}{\phi(M_0, M_1)} & \text{if } \phi(M_0, M_1) \neq 0; \\ 1 & \text{if } \phi(M_0, M_1) = \phi(M_1, M_0) = 0; \\ \infty & \text{otherwise.} \end{cases}$$

Let $\mathcal{M}$ denote the set of all matchings in G. The *unpopularity factor* of M is

$$u(M) = \max_{M' \in \mathcal{M}} \Delta(M, M').$$

Observe that by definition, $\Delta(M, M) = 1$. Hence for any matching M, $u(M) \geq 1$. In particular, M is popular if and only if $u(M) = 1$. In general, a matching M with a low value of $u(M)$ can be considered a good matching because $\phi(M', M) \leq u(M) \cdot \phi(M, M')$ for all M'; hence when comparing M with any matching M', the number of vertices that prefer M' to M cannot be larger than the number of other vertices by a factor of more than $u(M)$.

Least unpopularity factor matchings. Among all the matchings in $\mathcal{M}$, the one with the least value of unpopularity factor is called a *least unpopularity factor* matching. When G admits popular matchings, it is easy to see that every popular matching is a least unpopularity factor matching. However unlike popular matchings, least unpopularity factor matchings always exist. In the instance H on vertex set $\{a_0, a_1, a_2\}$ described earlier, M_0, M_1, and M_2 are least unpopularity factor matchings: the unpopularity factor of each of these matchings is 2 and the only other matching in H is the empty matching, whose unpopularity factor is ∞.

Since least unpopularity factor matchings are a generalization of popular matchings and because they always exist, a least unpopularity matching is a promising candidate for an optimal matching in G. However finding such a matching is APX-hard, as shown by our following result.

Theorem 1. *It is NP-hard to find a least unpopularity factor matching in a roommates instance $G = (V, E)$. In fact, for any $\epsilon > 0$, it is NP-hard to compute a matching whose unpopularity factor is at most $4/3 - \epsilon$ of the optimal. These hardness results hold even in the special case when G is a complete graph.*

Nevertheless, there is always a matching whose unpopularity factor is $O(\log n)$ and this can be computed efficiently, as shown by our following result.

Theorem 2. *Let G be a roommates instance on n vertices and m edges. Then G always admits a matching whose unpopularity factor is at most $4 \log n + O(1)$ and such a matching can be computed in $O(m + n)$ time.*

Thus though popular matchings do not always exist in the roommates instance, there is always a solution that is *not very unpopular* and it is efficiently computable. Thus we propose a solution that is reasonably optimal and efficiently computable for the problem of finding a good matching in a roommates instance. We show an instance $G = (V, E)$ where every matching has unpopularity factor $\Omega(\log |V|)$, hence the upper bound in Theorem 2 cannot be improved to $o(\log n)$.

Our algorithm can be also regarded an $O(\log |V|)$ approximation algorithm for finding the least unpopularity factor matching. We show that this approximation ratio cannot be improved to $o(\log |V|)$ by better analysis, since we construct an example in which our algorithm produces a matching whose unpopularity factor is $\Omega(\log |V|)$ while there exists a matching whose unpopularity factor is $O(1)$.

Popular matchings have been well-studied during the last few years [1, 6, 8–13]. Much of this work is in bipartite graphs where only vertices on one side (called *applicants*) have preferences while vertices on the other side (called *jobs*) have no preferences. So when we compare two matchings with respect to popularity, it is only applicants that cast their votes. This is called the one-sided preference lists model and popular matchings need not always exist here. Also, there exist simple instances here where every matching has unpopularity factor $\Omega(n)$: for instance, $A = \{a_1, \ldots, a_n\}$ and $B = \{b_1, \ldots, b_n\}$ where each $a \in A$ has the same preference list, which is, b_1 followed by b_2 followed by b_3 and so on till b_n as the last choice - every perfect matching here has unpopularity factor $n - 1$ (and non-perfect matchings have unpopularity factor ∞). Thus the existence of an $O(\log n)$ unpopularity factor matching in the roommates problem is surprising.

Background and Related Results. Gale and Shapley [3] introduced the stable marriage problem and as mentioned earlier, Irving [7] showed an efficient algorithm for determining if a roommates problem admits a stable matching or not and if so, to compute one. Gärdenfors [4] introduced the notion of popularity in the stable marriage problem. When ties are allowed in preference lists here, it has been shown by Biró, Irving, and Manlove [2] that the problem of computing an arbitrary popular matching is NP-hard.

For one-sided preference lists, there are efficient algorithms known for determining if the input instance admits a popular matching or not, and if so, to compute one [1]. McCutchen [12] defined two measures of unpopularity called the unpopularity factor and unpopularity margin and showed that in the one-sided

preference lists model, the problem of computing a matching that minimized either of these two measures is NP-hard. In bipartite graphs with two-sided strict preference lists, the problem of computing a least unpopularity factor/margin matching becomes easy due to the existence of stable matchings. Our hardness result shows that the hardness returns when we generalize to non-bipartite graphs.

Organization of the paper. Section 2 contains our linear time algorithm to construct a matching whose unpopularity factor is $O(\log n)$. Section 3 has our hardness result. In the Appendix, we show two instances: one where every matching has unpopularity factor $\Omega(\log |V|)$ and another which admits a matching of unpopularity factor $O(1)$, however our algorithm returns a matching of unpopularity factor $\Theta(\log |V|)$.

2 A low unpopularity factor matching

Let $G = (V, E)$ be an instance of the roommates problem. Tan [14] showed that, whether G admits a stable matching or not, G always admits a *stable partition*, which generalizes the notion of a stable matching. Note that a stable matching S can be regarded as a partition of V into sets, each of which has size either 2 (a pair matched to each other in S) or 1 (an unmatched vertex).

A stable partition is a partition $\{A_1, \ldots, A_k\}$ of the vertex set V, where each A_i is an *ordered set* $\langle a_i^0, \ldots, a_i^\ell \rangle$. We call a_i^{j-1} the *predecessor* of a_i^j and for every i and j, if $a_i^{j+1} \neq a_i^{j-1}$, then a_i^j prefers a_i^{j+1} (its *successor*) to a_i^{j-1}, where all superscripts are modulo $|A_i|$. When $A_i = \langle a_i^0 \rangle$, i.e., if A_i is a singleton set, we say that a_i^0 has no predecessor. Note that any cyclic permutation of A_i is also an ordered set. For every edge (a, b) in G, the following stable condition holds:

$(*)$ If a prefers b to its predecessor, then b does not prefer a to its predecessor.

If $A_i = \langle a_i^0 \rangle$, then a_i prefers any neighbor to its predecessor (which is non-existent) since a_i being matched to its predecessor is the same as a_i being left unmatched and this is the least preferred state for any vertex. Thus no neighbor of a_i can belong to a singleton set in a stable partition as that would contradict the stable condition. By this observation, for any $(a, b) \in E$, either a is b's predecessor/vice-versa or one of $\{a, b\}$ strictly prefers its predecessor to the other. A stable partition for the graph H on $\{a_0, \ldots, a_3\}$ described in Section 1 is $\{\langle a_0, a_1, a_2 \rangle, \langle a_3 \rangle\}$.

We will use the following notation: for any vertex u and neighbors v and w of u, $\mathsf{vote}_u(v, w)$ is 1 if u prefers v to w, it is -1 if u prefers w to v, and it is 0 otherwise (i.e., $v = w$). Also, if matching M leaves u unmatched, then $\mathsf{vote}_u(v, M(u)) = 1$ where v is any neighbor of u.

2.1 Our algorithm

We present below an algorithm for finding an $O(\log |V|)$ unpopularity factor matching in $G = (V, E)$. Our input is the graph $G_0 = G$ (so the vertex set V_0 refers to V).

1. Let $\mathcal{P}_0$ be a stable partition $\{A_1, \ldots, A_k\}$ of G_0.
   1.1 Set $X_0 = \cup_{i=1}^{k} \{a_i^{2t-1} : t = 1, \ldots, \lceil |A_i|/2 \rceil\}$, that is, X_0 is the set of all *odd* indexed vertices in all the ordered sets in the partition $\mathcal{P}_0$.
   {*recall that vertices in the ordered set* A_i *are indexed* $\langle a_i^0, a_i^1, \ldots, a_i^{|A_i|-1} \rangle$}
   1.2 Run the Gale-Shapley algorithm on $(X_0, V_0 \setminus X_0)$.
   {*vertices of* X_0 *propose to those in* $V_0 \setminus X_0$ *and vertices of* $V_0 \setminus X_0$ *dispose*}
   Let M_0 denote the resulting matching and let Y_0 denote the set of matched vertices in $V_0 \setminus X_0$.
2. Let V_1 denote the set of *unmatched* vertices in $V_0 \setminus X_0$. In other words, $V_1 = V_0 \setminus (X_0 \cup Y_0)$.
   2.1 Let G_1 be the induced subgraph on V_1. Delete all isolated vertices from G_1 (and V_1).
   2.2 If $V_1 = \emptyset$, then return $S = M_0$. Else let S' be the matching returned by running our algorithm recursively on G_1. Return $S = M_0 \cup S'$.

Note that the Gale-Shapley algorithm is described in the proof of Lemma 1. In our algorithm above, just as V_0 was partitioned into X_0, Y_0, V_1, in the recursive call for G_1, the vertex set V_1 gets partitioned into X_1, Y_1, V_2, and in the recursive call for G_2 (the induced subgraph on V_2), the vertex set V_2 gets partitioned into X_2, Y_2, V_3, and so on till V_{r+1} is empty, where r is the recursion depth in our algorithm. We also add all isolated vertices pruned from V_i in Step 2.1, for $i = 1, \ldots, r+1$, to V_{r+1}. Let $X = X_0 \cup X_1 \cup \cdots \cup X_r$ and let $Y = Y_0 \cup Y_1 \cup \cdots \cup Y_r \cup V_{r+1}$. We will show that the following properties hold:

(I) For each $0 \leq i \leq r$, every vertex in X_i is matched to a vertex in Y_i and vice-versa. Every unmatched vertex has to be in V_{r+1}.

(II) If we label each edge $(u, v) \in E \setminus S$ by (α, β) where α is $\mathsf{vote}_u(v, S(u))$ and β is $\mathsf{vote}_v(u, S(v))$, then
- there is no edge labeled $(1, 1)$ between any two vertices in X, and
- there is no edge labeled $(1, 1)$ between any $x \in X_i$ and any $y \in Y_i \cup Y_{i+1} \cup \cdots \cup Y_r \cup V_{r+1}$.

Lemma 1. *For each i, every $x \in X_i$ gets matched in S to a vertex that x considers at least as good as its predecessor in the stable partition $\mathcal{P}_i$ of G_i.*

Proof. Recall the Gale-Shapley algorithm on the edge set restricted to $X_i \times (V_i \setminus X_i)$ when vertices of X_i propose and vertices of $V_i \setminus X_i$ dispose. We assume that all the unmatched vertices of X_i are placed in a queue. While there exists a vertex x in the queue that has not yet been rejected by all its neighbors, x is removed from the queue and it proposes to its most preferred neighbor y in $V_i \setminus X_i$ that has not yet rejected x. If y is unmatched or prefers x to its current neighbor z, then x gets matched to y; else x is rejected by y. If y was already matched to some vertex z when x proposes to y, then the vertex who is rejected by y (either x or z) gets added to the queue of unmatched vertices. This goes on till every $x \in X_i$ is either matched or has been rejected by all its neighbors in $V_i \setminus X_i$.

For each $x \in X_i$, let $p_i(x)$ denote the predecessor of x in the stable partition $\mathcal{P}_i$ of G_i. We need to show that each $x \in X_i$ gets matched to a vertex y that x considers at least as good as $p_i(x)$. Suppose not. Let $z \in X_i$ be the first vertex in the Gale-Shapley algorithm that gets rejected by $p_i(z)$. That is, the vertex $p_i(z)$ is matched to a vertex w that $p_i(z)$ prefers to z. Since z was the *first* vertex that is rejected by its predecessor in the Gale-Shapley algorithm, it has to be the case that w prefers $p_i(z)$ to $p_i(w)$. Also, $p_i(z)$ prefers w to its predecessor - this is because $p_i(z)$ prefers w to its successor z (recall that in any ordered set, every vertex likes its successor at least as much as its predecessor). The fact that both w and $p_i(z)$ prefer each other to their respective predecessors contradicts the stable property $(*)$ stated at the beginning of Section 2. Hence there cannot be any vertex z during the entire course of the Gale-Shapley algorithm that gets rejected by its predecessor. The lemma follows. □

It follows from Lemma 1 that every $x \in X_i$ is matched in S. For each $0 \leq i \leq r$, let M_i be the matching S restricted to $X_i \cup Y_i$. Since $Y_i \subseteq V_i \setminus X_i$ is the set of matched vertices in M_i, we can conclude the following corollary.

Corollary 1. *For each $0 \leq i \leq r$, M_i is a perfect matching on $X_i \cup Y_i$.*

Hence it follows that every unmatched vertex has to be in $V \setminus (\cup_{i=0}^{r} X_i \cup_{i=0}^{r} Y_i)$, which is V_{r+1}. Thus we have proved property (I). Now we show property (II).

Lemma 2. *With respect to the matching S,*

(1) there is no edge labeled $(1, 1)$ between any pair of vertices in X, and
(2) for every $0 \leq i \leq r$, there is no edge labeled $(1, 1)$ between a vertex $x \in X_i$ and a vertex $y \in Y_i \cup Y_{i+1} \cup \cdots \cup Y_r \cup V_{r+1}$.

Proof. Lemma 1 tells us that each vertex $x \in X_0$ gets matched to a vertex that is at least as good as its predecessor in $\mathcal{P}_0$. So property $(*)$ of a stable partition implies that there can be no $(1, 1)$ edge between any two vertices of X_0. Since we run Gale-Shapley algorithm between $(X_0, V \setminus X_0)$, there can be no $(1, 1)$ edge between a vertex of X_0 and a vertex of $V \setminus X_0$. Thus it follows that there can be no $(1, 1)$ edge between an $x_0 \in X_0$ and any vertex of V. Hence there is no $(1, 1)$ edge between an $x_0 \in X_0$ and any $x \in X$; also there is no $(1, 1)$ edge between an $x_0 \in X_0$ and a vertex $y \in Y_0 \cup \cdots \cup Y_r \cup V_{r+1}$.

Applying the same argument in G_1, we see that there is no $(1, 1)$ edge between an $x_1 \in X_1$ and any vertex of V_1, i.e., there is no $(1, 1)$ edge between an $x_1 \in X_1$ and any vertex of $X_1 \cup \cdots X_r$; also there is no $(1, 1)$ edge between an $x_1 \in X_1$ and a vertex $y \in Y_1 \cup \cdots \cup Y_r \cup V_{r+1}$.

Continuing this argument, we see that there is no $(1, 1)$ edge between an $x_i \in X_i$ and any vertex of $X_i \cup \cdots X_r$; also there is no $(1, 1)$ edge between an $x_i \in X_i$ and a vertex $y \in Y_i \cup \cdots \cup Y_r \cup V_{r+1}$.

Thus there is no $(1, 1)$ edge between any pair of vertices in X and for every $0 \leq i \leq r$, there is no $(1, 1)$ edge between a vertex $x \in X_i$ and a vertex $y \in Y_i \cup Y_{i+1} \cup \cdots \cup Y_r \cup V_{r+1}$. □

The properties of the matching S (as given by Lemmas 1 and 2) enable us to show Lemma 3. We say an alternating path/cycle ρ with respect to S has k *consecutive* $(1,1)$ edges if ρ has a subpath v_0-v_1-$S(v_1)$-$v_2 \cdots S(v_{k-1})$-v_k where every unmatched edge (note that there are k of them) is labeled $(1,1)$.

Lemma 3. *There can be at most $2r+1$ consecutive $(1,1)$ edges in any alternating path/cycle with respect to S. No alternating cycle can consist solely of $(1,1)$ edges.*

Proof. Let ρ be an alternating path with respect to S and let ρ' be the longest subpath of consecutive $(1,1)$ edges in ρ. Since it is only edges of $E \setminus S$ that get labeled by the votes of their endpoints, we can assume that the first edge of ρ' is an unmatched edge. Let u_0 be an endpoint of ρ'. There are two cases: (i) $u_0 \in X$, (ii) $u_0 \in Y$.

Case (i). Let u_0 be in X_i. Since every unmatched edge of ρ' is marked $(1,1)$, it has to be the case that the vertex that follows u_0 in ρ', call this vertex v_0, has to be in $Y_{i-1} \cup Y_{i-2} \cup \cdots \cup Y_0$, since there are no $(1,1)$ edges in $X_i \times (X \cup Y_i \cup \cdots \cup V_{r+1})$. Suppose $v_0 \in Y_j$, where $0 \leq j \leq i-1$. Then $S(v_0) \in X_j$. Let u_1 be the vertex $S(v_0)$. It follows from the same argument that the vertex after u_1 in ρ', call this vertex v_1, has to be in $Y_{j-1} \cup Y_{j-2} \cup \cdots \cup Y_0$. Suppose $v_1 \in Y_k$, where $0 \leq k \leq j-1$. Then $S(v_1) \in X_k$. However we cannot continue in this manner for more than r edges since we will be at a vertex in X_0 after at most r such edges and there are no $(1,1)$ edges incident on any vertex in X_0. Thus ρ' has at most r consecutive $(1,1)$ edges in this case.

Case (ii). Let u_0 be in Y_i. The vertex v_0 succeeding u_0 in ρ' can be either in X or in Y. If $v_0 \in Y$, then $u_1 = S(v_0)$ has to be in X and this becomes exactly the same as Case (i) and so there can be at most r consecutive $(1,1)$ edges after u_1. So let us assume that $v_0 \in X$. Since the edge (u_0, v_0) is labeled $(1,1)$ and there are no $(1,1)$ edges between Y_i and $X_i \cup X_{i-1} \cup \cdots \cup X_0$, the vertex v_0 has to be in X_j, where $j \geq i+1$. Hence $u_1 = S(v_0)$ is in Y_j. Again, the vertex v_1 that follows u_1 in ρ' is either in X or in Y. If $v_1 \in Y$, then this again becomes exactly the same as Case (i). Hence we assume that $v_1 \in X$. Since (u_1, v_1) is labeled $(1,1)$, it follows that v_1 has to be in X_k, where $k \geq j+1$. We cannot see more than r such $(1,1)$ edges of $Y \times X$ in ρ', since after seeing at most r such edges, we reach a vertex in Y_r (call this vertex u_ℓ) and there are no $(1,1)$ edges between any vertex in Y_r and a vertex in X. Hence the vertex v_ℓ that follows u_ℓ is also in Y. Then the vertex $u_{\ell+1} = S(v_\ell)$ is in X and the same argument as in Case (i) goes through, and so we have at most r consecutive $(1,1)$ edges in ρ' after we reach $u_{\ell+1} \in X$. Thus the total number of $(1,1)$ edges in ρ' is at most r (from u_0 to u_ℓ) $+ 1$ (for the edge (u_ℓ, v_ℓ)) $+ r$ (from $u_{\ell+1}$ onwards), which adds up to $2r+1$.

Let ρ be an alternating cycle. First, we prove that not every non-matching edge in ρ can be a $(1,1)$ edge. Pick any vertex $u_0 \in X$ as our starting vertex in ρ. The same argument as in Case (i) holds and if we traverse r consecutive $(1,1)$ edges starting from u_0, then we have to be at a vertex u_ℓ in X_0. As there are no $(1,1)$ edges incident on any vertex in X_0, we have to see an edge labeled $(-1,1)$, $(1,-1)$, or $(-1,-1)$ after reaching u_ℓ. Thus there cannot be an alternating cycle with only $(1,1)$ edges. Let ρ' be a subpath of ρ that consists of only $(1,1)$ edges. Now the same proof as above (when ρ was an alternating path) holds here too: thus the total number of $(1,1)$ edges in ρ' is at most $2r+1$. □

Lemma 3 leads to Lemma 4 that bounds $u(S)$ in terms of r. Lemma 5 bounds r in terms of $|V|$.

Lemma 4. $u(S) \leq 4r + O(1)$.

Proof. Let M be any matching. It is easy to see that $\Delta(S, M)$ is upper bounded by $\max_\rho \Delta(S, S \oplus \rho)$, where $\rho \in S \oplus M$. So we bound $\Delta(S, S \oplus \rho)$ for all $\rho \in S \oplus M$ by $4r + O(1)$ now.

Suppose ρ is an alternating cycle. Lemma 3 tells us that after seeing $2r+1$ consecutive $(1,1)$ edges in ρ, we have to see an edge marked (α, β) where either α or β (or both) is -1. Thus sandwiched between 2 non-$(1,1)$ edges, we can have at most $2r+1$ consecutive $(1,1)$ edges. Thus in the list $\langle \mathsf{vote}_u(M(u), S(u)) \rangle$ where $u \in \rho$, we can assign to every -1, at most $4r+3$ 1's. This implies that $\Delta(S, S \oplus \rho) \leq 4r+3$.

Suppose ρ is an alternating path. If ρ has an endpoint unmatched in S, then that vertex (call it u) has to be in V_{r+1}. It follows from Case (ii) of the proof of Lemma 3 that we can have at most $r+1$ consecutive $(1,1)$ edges starting from u. Hence if we look at the list $\langle \mathsf{vote}_u(M(u), S(u)) \rangle$ for all $u \in \rho$, we can have at most $2r+2$ consecutive 1's as a prefix (similarly, suffix), and thereafter there can be at most $4r+4$ 1's sandwiched between 2 -1's. Thus $\Delta(S, S \oplus \rho)$ is maximized by assuming ρ to be a prefix of $r+1$ consecutive

$(1,1)$ edges, followed by a $(-1,1)$ edge, and followed by a suffix of $r+1$ consecutive $(1,1)$ edges. Hence $\Delta(S, S \oplus \rho) \leq 4r+5$.

Hence it follows that $\max_\rho \Delta(S, S \oplus \rho)$ is at most $4r + O(1)$. Thus $\Delta(S, M)$ for any matching M is at most $4r + O(1)$, in other words, $u(S) \leq 4r + O(1)$. □

Lemma 5. *The recursion depth in our algorithm, i.e., r, is at most* $\log |V|$.

Proof. For any i, where $0 \leq i \leq r$, let $\mathcal{P}_i$ be the stable partition of the graph G_i computed in our algorithm. Let o_i be the number of odd cardinality ordered sets in $\mathcal{P}_i$. Since $|M_i| = 2|X_i|$ where X_i includes exactly $\lfloor |A_j|/2 \rfloor$ vertices from each ordered set A_j in $\mathcal{P}_i$, it follows that the number of unmatched vertices in M_i is o_i. That is, the size of V_{i+1}, before isolated vertices are deleted from V_{i+1}, is exactly o_i.

All vertices that formed singleton sets in $\mathcal{P}_i$ are in V_{i+1} before we delete isolated vertices from V_{i+1}. Let U_i denote the set of vertices that formed singleton sets in $\mathcal{P}_i$. From the definition of a stable partition, it follows that U_i has to form an independent set in G_i. Thus the size of a minimum cardinality vertex cover in G_{i+1} is at most $o_i - |U_i|$, which is the number of odd cardinality ordered sets of size ≥ 3 in $\mathcal{P}_i$.

Let C_i be a vertex cover of G_i. Since C_i has to include at least 2 vertices from every odd cardinality ordered set of size ≥ 3 in $\mathcal{P}_i$, we have $|C_i| \geq 2(o_i - |U_i|)$. Thus $c_i \geq 2c_{i+1}$, where c_i (similarly, c_{i+1}) is the size of a minimum cardinality vertex cover of G_i (resp., G_{i+1}). This inequality holds for every $0 \leq i \leq r$. Since the edge set of G_r is non-empty, $c_r \geq 1$. Thus we get $r \leq \log c_0 \leq \log |V|$. □

It follows from Lemmas 4 and 5 that $u(S) \leq 4 \log |V| + O(1)$, and the first part of Theorem 2 is proved. We next discuss how to bound the running time of our algorithm so as to prove the second part of Theorem 2.

Running time of the algorithm The running time of the algorithm in the i-th recursive call is $O(|V_i| + |E_i|)$, where V_i and E_i are the sets of vertices and edges in G_i, respectively. Since all isolated vertices get deleted from the vertex set V_i, it follows that $|E_i| \geq |V_i|/2$. Hence the running time of our algorithm is $O(n + \sum_{i=0}^{r} |E_i|)$.

For each $0 \leq i \leq r$, we would like to upper bound $|E_{i+1}|$ by $2|E_i|/3$. Let $A_j = \langle a_j^1, \ldots, a_j^t \rangle$ be an ordered set of cardinality at least 2 in $\mathcal{P}_i$. Let m_j be the total number of edges incident on vertices of A_j in G_i. We would like to ensure that at least $1/3$ of these m_j edges have one or more endpoints in the set of all the *odd* indexed vertices in A_j. If this property holds for each A_j such that $|A_j| \geq 2$, because the singleton sets in $\mathcal{P}_i$ form an independent set, it follows that the number of edges in G_{i+1} is at most $\frac{2}{3}$(the number of edges in G_i).

If in the current order in A_j, all the odd indexed vertices in A_j have less than $m_j/3$ edges incident on them, then we will perform a clockwise rotation on A_j, call this ordered set $\sigma(A_j)$, and also a counter clockwise rotation on A_j, call this ordered set $\sigma^{-1}(A_j)$. If $|A_j|$ is even, then all the even indexed vertices of A_j become odd indexed vertices in $\sigma(A_j)$. If $|A_j|$ is odd, then we cannot make this claim; however, all the even indexed vertices of A_j become odd indexed vertices in either $\sigma(A_j)$ or $\sigma^{-1}(A_j)$. Thus all the odd indexed vertices in one of A_j, $\sigma(A_j)$, $\sigma^{-1}(A_j)$ have at least $m_j/3$ edges incident on them.

So in the recursive call for G_i, after we obtain $\mathcal{P}_i$, for each A_j, we either leave it as it is or perform a clockwise/counter clockwise rotation on it and call the resulting ordered set A_j so that all the odd indexed vertices in A_j have at least $m_j/3$ edges incident on them. Thus the running time of our algorithm, which is $O(n + \sum_{i=0}^{r} |E_i|)$, becomes $O(n + \sum_{i=0}^{r} (\frac{2}{3})^i m))$, which is $O(n+m)$. This completes the proof of Theorem 2.

As mentioned in Section 1, we show the following instances in the Appendix:

(1) an instance where every matching has unpopularity factor at least $\Omega(\log |V|)$—therefore the upper bound as stated in Theorem 2 cannot be further improved
(2) an instance where our algorithm returns a matching of unpopularity factor $\Theta(\log |V|)$ while there exists a matching of unpopularity factor $O(1)$. Thus the approximation guarantee of our algorithm (when viewed as an approximation algorithm) cannot be improved to a bound better than $O(\log |V|)$, by better analysis.

3 Least unpopularity factor matching

In this section, we prove Theorem 1 by presenting a reduction from 1-IN-3 SAT. In 1-IN-3 SAT, a formula $\phi = C_1 \wedge C_2 \wedge \cdots \wedge C_m$ in conjunctive normal form is given, where each clause $C_j = x_1^j \vee x_2^j \vee x_3^j$ is a disjunction of three non-negated literals. The formula ϕ is satisfiable iff there exists an assignment where exactly one literal in each clause is set to true. This decision problem is NP-complete [5].

Our reduction will construct a roommates instance G_ϕ such that if ϕ is a *yes* instance for 1-IN-3 SAT, then there exists a matching M with $u(M) \leq 1.5$, and if ϕ is a *no* instance, then every matching M has $u(M) \geq 2$. Also, G_ϕ will be a complete graph.

To avoid confusion, we use the upper case X_i to refer to a variable, while the lower case x_t^j means the t-th literal in clause C_j. For instance, if $C_j = (X_1 \vee X_5 \vee X_{10})$, then $x_1^j = X_1$, $x_2^j = X_5$, and $x_3^j = X_{10}$.

We have two types of gadgets: variable gadget and clause gadget. For each variable X_i, we create 16 vertices that form a variable gadget and for each clause C_j, we create 20 vertices that form a clause gadget. Note that since the preferences are complete and the number of vertices is even, if a matching is not perfect, then its unpopularity factor will be ∞. In the following discussion, we implicitly assume that a matching is perfect.

3.1 Variable Gadget

For each variable X_i, we create 16 vertices: $a_1, \ldots, a_7$ and $b_1, \ldots, b_9$. Their preference lists are shown in Table 1. The function $\pi(\cdot)$ is an arbitrary permutation of some vertices from clause gadgets and we will explain who they are later. The "$\cdots$" is an arbitrary permutation of all the remaining vertices.

Table 1. The preference lists of the vertices in the variable gadget for X_i.

Vertex	Preference List	Vertex	Preference List
a_1^i	$a_2^i\ b_1^i\ a_7^i\ \cdots$	b_1^i	$b_2^i\ b_9^i\ \pi(b_1^i)\ a_1^i\ \cdots$
a_2^i	$a_3^i\ a_1^i\ \cdots$	b_2^i	$b_3^i\ b_1^i\ \cdots$
a_3^i	$a_4^i\ \pi(a_3^i)\ a_2^i\ \cdots$	b_3^i	$b_4^i\ b_2^i\ \cdots$
a_4^i	$a_5^i\ a_3^i\ \cdots$	b_4^i	$b_5^i\ b_3^i\ \cdots$
a_5^i	$a_6^i\ b_9^i\ a_4^i\ \cdots$	b_5^i	$b_6^i\ b_4^i\ \cdots$
a_6^i	$a_7^i\ a_5^i\ \cdots$	b_6^i	$b_7^i\ b_5^i\ \cdots$
a_7^i	$a_1^i\ \pi(a_7^i)\ a_6^i\ \cdots$	b_7^i	$b_8^i\ b_6^i\ \cdots$
		b_8^i	$b_9^i\ b_7^i\ \cdots$
		b_9^i	$b_1^i\ b_8^i\ \pi(b_9^i)\ a_5^i \cdots$

We will define the function $\pi(\cdot)$ in such as way so that the following holds.

Proposition 1 *In any matching M in G_ϕ, if $u(M) < 2$, then for every i:*

(i) b_k^i is not matched to any vertex in $\pi(b_k^i)$ for $k = 1$ and $k = 9$.
(ii) None of the vertices in $\{a_t^i\}_{t=1}^7 \cup \{b_t^i\}_{t=1}^9$ is matched to any vertex in the "$\cdots$" part of their preference lists.

We will show Proposition 1 holds after we finish the description of G_ϕ. For now, we assume it is true and show Lemma 6.

Lemma 6. *If there is a matching M with $u(M) < 2$, then the vertices corresponding to variable X_i can only be matched in one of the following two ways.*

(i) (a_1^i, b_1^i), (a_2^i, a_3^i), (a_4^i, a_5^i), (a_6^i, a_7^i), (b_2^i, b_3^i), (b_4^i, b_5^i), (b_6^i, b_7^i), $(b_8^i, b_9^i) \in M$—in this case, we say the variable X_i is set to true.
(ii) (a_1^i, a_2^i), (a_3^i, a_4^i), (a_5^i, b_9^i), (a_6^i, a_7^i), (b_1^i, b_2^i), (b_3^i, b_4^i), (b_5^i, b_6^i), $(b_7^i, b_8^i) \in M$—in this case, we say the variable X_i is set to false.

Proof. We cannot have M match all the vertices in $\{b_1^i, \ldots, b_9^i\}$ among themselves, since there are an odd number of these vertices. Since $u(M) < 2$, by Proposition 1, it follows that either b_1^i is matched to a_1^i, or b_9^i is matched to a_5^i, but not both, otherwise, some vertices in $\{b_t^i\}_{t=2}^8$ would have to be matched to the vertices in the "$\cdots$" part in their lists, contradicting Proposition 1.

Now if $(a_1^i, b_1^i) \in M$, it is easy to see that (i) is the only possible way to match these vertices so as to maintain the property that $u(M) < 2$. The same applies to (ii) if $(a_5^i, b_9^i) \in M$. □

3.2 Clause Gadget

For each clause $C_j = x_1^j \vee x_2^j \vee x_3^j$, we create 20 vertices: $c_1^j, c_2^j, c_3^j, d_1^j, \ldots, d_{17}^j$. The preference list for 14 of the vertices is given below:

$$d_t^j : \quad d_{t+1}^j \quad d_{t-1}^j \quad \cdots \qquad \text{for } t \in \{2, \ldots, 8\} \cup \{10, \ldots, 16\}.$$

The "$\cdots$" is an arbitrary permutation of those remaining vertices not explicitly listed. The preference lists of the other vertices are shown in Table 2. As before, $\pi(\cdot)$ stands for an arbitrary permutation of some vertices from variable gadgets and we will explain who they are later.

Table 2. The preference lists of six vertices in the clause gadget for $C_j = x_1^j \vee x_2^j \vee x_3^j$.

Vertex	Preference List	Vertex	Preference List
c_1^j	$c_2^j\ d_1^j\ c_3^j\ \pi(c_1^j)\ \cdots$	d_1^j	$d_2^j\ d_{17}^j\ \pi(d_1^j)\ c_1^j\ \cdots$
c_2^j	$c_3^j\ d_9^j\ c_1^j\ \pi(c_2^j)\ \cdots$	d_9^j	$d_{10}^j\ d_8^j\ \pi(d_9^j)\ c_2^j\ \cdots$
c_3^j	$c_1^j\ d_{17}^j\ c_2^j\ \pi(c_3^j)\ \cdots$	d_{17}^j	$d_1^j\ d_{16}^j\ \pi(d_{17}^j)\ c_3^j\ \cdots$

We will define the function $\pi(\cdot)$ in such as way so that the following holds.

Proposition 2 *In any matching M in G_ϕ, if $u(M) < 2$, then for every j:*

(i) d_k^j is not matched to any vertex in $\pi(d_k^j)$ for $k = 1, 9$, and 17.
(ii) None of the vertices in $\{c_t^j\}_{t=1}^3 \cup \{d_t^j\}_{t=1}^{17}$ is matched to any vertex in the "$\cdots$" part of their preference lists.

We will show Proposition 2 holds after we finish the description of G_ϕ. For now, we assume it is true and show Lemma 7.

Lemma 7. *If there is a matching M such that $u(M) < 2$, then the vertices corresponding to clause $C_j = x_1^j \vee x_2^j \vee x_3^j$ can only be matched in one of the following three ways:*

(i) (c_1^j, d_1^j), (c_2^j, c_3^j), (d_{2k}^j, d_{2k+1}^j), for $1 \le k \le 8$, are in M —in this case we say the first literal x_1^j is set to true.
(ii) (c_2^j, d_9^j), (c_1^j, c_3^j), (d_{2k-1}^j, d_{2k}^j), for $1 \le k \le 4$, (d_{2k}^j, d_{2k+1}^j), for $5 \le k \le 8$, are in M —in this case we say the second literal x_2^j is set to true.
(iii) (c_3^j, d_{17}^j), (c_1^j, c_2^j), (d_{2k-1}^j, d_{2k}^j), for $1 \le k \le 8$, are in M —in this case we say the third literal x_3^j is set to true.

Proof. If $u(M) < 2$, then by Proposition 2, exactly one of the following three edges can be in M: (c_1^j, d_1^j), (c_2^j, d_9^j), (c_3^j, d_{17}^j), otherwise, some vertices in $\{d_t^j\}_{t=2}^8 \cup \{d_t^j\}_{t=10}^{16}$ would have to be matched to the vertices in the "$\cdots$" part in their lists, contradicting Proposition 2.

Now if $(c_1^j, d_1^j) \in M$, it is easy to see that (i) is the only possible way to match all the vertices so as to maintain the property that $u(M) < 2$. The same applies to (ii) if $(c_2^j, d_9^j) \in M$ and to (iii) if $(c_3^j, d_{17}^j) \in M$. □

3.3 How the two types of gadgets interact

We now explain how the two types of gadgets work together by specifying the function π. It may be helpful to first use a simple example to illustrate our ideas. Suppose $C_1 = (X_1 \vee X_5 \vee X_{10})$. Intuitively, we want the following when the derived instance has a matching M with $u(M) < 2$: if the first literal of C_1 is set to true (i.e., $(c_1^1, d_1^1) \in M$—see Lemma 7), then we want to make sure that X_1 is set to true while X_5 and X_{10} are set to false (i.e., we want (a_1^1, b_1^1), (a_5^5, b_9^5), and (a_5^{10}, b_9^{10}) part of M—see Lemma 6.)

Our construction of the function π makes sure if the assignment is "inconsistent", for instance, the first literal $x_1^j = X_1$ of C_1 is set to true but the variable X_1 itself is set to false, i.e., if both (c_1^1, d_1^1) and (a_5^1, b_9^1) are in M, then we can find an alternating cycle with a $(1,1)$ and two $(1,-1)$ edges, where every edge in $(u,v) \in E \setminus M$ is labeled $(\mathsf{vote}_u(v, M(u)), \mathsf{vote}_v(u, M(v)))$. This would cause M to have unpopularity factor at least 2. Specifically, we define $\pi(\cdot)$ as follows.

1. For all i and j: a_3^i, a_7^i are in $\pi(c_1^j)$, in $\pi(c_2^j)$, and in $\pi(c_3^j)$. Symmetrically, c_1^j, c_2^j, c_3^j are in $\pi(a_3^i)$ and in $\pi(a_7^i)$.
2. For each j, we ensure the following inclusions: suppose $C_j = x_1^j \vee x_2^j \vee x_3^j$ and $X_i = x_1^j$, $X_k = x_2^j$, $X_t = x_3^j$. Then
   (a) b_9^i, b_1^k, b_1^t are in $\pi(d_1^j)$; symmetrically, d_1^j is in $\pi(b_9^i), \pi(b_1^k)$, and $\pi(b_1^t)$.
   (b) b_1^i, b_9^k, b_1^t are in $\pi(d_9^j)$; symmetrically, d_9^j is in $\pi(b_1^i), \pi(b_9^k)$, and $\pi(b_1^t)$.
   (c) b_1^i, b_1^k, b_9^t are in $\pi(d_{17}^j)$; symmetrically, d_{17}^j is in $\pi(b_1^i), \pi(b_1^k)$, and $\pi(b_9^t)$.

[Observe that the function π is symmetrical. If a vertex $c \in \pi(a)$, then $a \in \pi(c)$; if a vertex $b \in \pi(d)$, then $d \in \pi(b)$. Moreover, our construction ensures that β belongs to the "$\cdots$" part of α's list if and only is α belongs to the "$\cdots$" part of β's list.]

To illustrate how the above definitions of $\pi(\cdot)$ help us achieve consistency, consider the above example. Suppose that (c_1^1, d_1^1), (a_5^1, b_9^1) are in M. Consider the alternating cycle $\rho = c_1^1$-d_1^1-b_9^1-a_5^1-a_6^1-a_7^1-c_1^1. $\Delta(M, M \oplus \rho) = 4/2$ since d_1^1, b_9^1, a_5^1, and a_7^1 are better off in $M \oplus \rho$ while a_6^1 and c_1^1 are worse off. Thus $u(M) \geq 2$.

The construction of G_ϕ is complete and we now prove Propositions 1 and 2.

Proofs of Propositions 1 and 2. Let b_t^i be an element in $\pi(d_s^j)$, so d_s^j is also in $\pi(b_t^i)$. Suppose the edge $(b_t^i, d_s^j) \in M$. Then (b_t^i, b_{t-1}^i) and (d_s^j, d_{s-1}^j) are both $(1,1)$ edges, where b_0^i (similarly, d_0^j) is the same as b_9^i (resp., d_{17}^j).

The matching obtained by augmenting M along the alternating path $\rho = M(b_{t-1}^i)$-b_{t-1}^i-b_t^i-d_s^j-d_{s-1}^j-$M(d_{s-1}^j)$ makes b_t^i, b_{t-1}^i, d_s^j, and d_{s-1}^j better off while $M(b_{t-1}^i)$ and $M(d_{s-1}^j)$ are worse off. Thus $\Delta(M, M \oplus \rho) = 2$. Thus $u(M) \geq 2$ and we have proved (i) in both propositions.

To prove (ii) in both the propositions, assume that $(y, z) \in M$ and y and z list each other in the "$\cdots$" part of their preference lists.

- for $1 \leq t \leq 7$, if y (or z) is a_t^i, then y' (resp., z') is a_{t-1}^i (where a_0^i is a_7^i);
- for $1 \leq t \leq 9$, if y (or z) is b_t^i, then y' (resp., z') is b_{t-1}^i (where b_0^i is b_9^i);
- for $1 \leq t \leq 3$, if y (or z) is c_t^j, then y' (resp., z') is c_{t-1}^j (where c_0^j is c_3^j);
- for $1 \leq t \leq 17$, if y (or z) is d_t^j, then y' (resp., z') is d_{t-1}^j (where d_0^j is d_{17}^j).

Consider the alternating path $\rho = M(y')$-y'-y-z-z'-$M(z')$. Augmenting M along ρ makes y, y', z, and z' better off while $M(y')$ and $M(z')$ are worse off. Thus $u(M) \geq 2$ and we have proved (ii) in both propositions. □

3.4 Correctness of Our Reduction

Lemma 8. *Suppose that there is a matching M with $u(M) < 2$ in G_ϕ. Then there exists a satisfying assignment to ϕ.*

Proof. We construct a truth assignment for ϕ based on M as follows. By Lemma 7, for each clause gadget of $C_j = x_1^j \vee x_2^j \vee x_3^j$, one of its three literals x_t^j is set to true. Set the variable X_i to true if $X_i = x_t^j$. X_i is set to false if it is never set to true. We claim that this yields a satisfying assignment for ϕ.

First note that at least one of the literals in each clause is set to true. So if we do not have a satisfying assignment, it must be the case that some clause $C_j = x_1^j \vee x_2^j \vee x_3^j$ has two (or more) literals being set to true. Without loss of generality, assume that $X_1 = x_1^j$ and $X_2 = x_2^j$ and X_1 is set to true because in the matching M, the first literal of C_j is satisfied, i.e., the edges (c_1^j, d_1^j), (c_2^j, c_3^j), (d_{2k}^j, d_{2k+1}^j), for $1 \leq k \leq 8$, are in M.

Then as X_2 is also set to true, there must exist another clause $C_t \neq C_j$ and $C_t = x_1^t \vee x_2^t \vee x_3^t$ and C_t is satisfied by its, say, first literal. So $x_1^t = X_2$ and (c_1^t, d_1^t), (c_2^t, c_3^t), (d_{2k}^t, d_{2k+1}^t), for $1 \leq k \leq 8$, are in M.

Now by Lemma 6, either (a_1^2, b_1^2) and (a_2^2, a_3^2) are in M, or (a_5^2, b_9^2) and (a_6^2, a_7^2) are in M. In the former case, augmenting M along the alternating cycle b_1^2-d_1^j-c_1^j-a_3^2-a_2^2-a_1^2-b_1^2 makes b_1^2, d_1^j ,a_3^2, and a_1^2 better off while c_1^j and a_2^2 are worse off; in the latter case, augmenting M along the alternating cycle b_9^2-d_1^t-c_1^t-a_7^2-a_6^2-a_5^2-b_9^2 makes b_9^2, d_1^t, a_7^2, and a_5^2 better off while c_1^t and a_6^2 are worse off. In both cases, we have $u(M) \geq 2$, a contradiction. □

Conversely, suppose ϕ is satisfiable. Then Lemma 9 constructs a matching M in G_ϕ, based on Lemmas 6 and 7, so that $u(M) \leq 1.5$.

Lemma 9. *Suppose that there is a satisfying assignment for ϕ. Then there is a matching M with $u(M) \leq 1.5$ in G_ϕ.*

We will first show that Theorem 1 follows from Lemmas 8 and 9. Then we will present the proof of Lemma 9.

Proof of Theorem 1. Given an input instance ϕ of 1-IN-3 SAT, we build the graph G_ϕ. Let M be a matching in G_ϕ whose unpopularity factor is strictly smaller than 4/3 of the least unpopularity factor matching.

- If $u(M) < 2$, then ϕ is a *yes* instance (as shown by Lemma 8).
- If $u(M) \geq 2$, then ϕ has to be a *no* instance. Otherwise, by Lemma 9, we know that there is a matching with unpopularity factor at most 1.5. This implies $u(M) < 2$ since $u(M)$ has to be smaller than 4/3 of the optimal.

Thus the problem of computing a matching whose unpopularity factor is strictly smaller than 4/3 of the optimal is NP-hard. □

Proof of Lemma 9 The formula ϕ is a satisfiable instance of 1-IN-3 SAT. We have to show a matching M in G_ϕ whose unpopularity factor is at most 1.5. Suppose ϕ has n variables and m clauses. Then $M = \cup_{i=1}^n T_i \cup_{j=1}^m T'_j$, where T_i for $1 \leq i \leq n$ and T'_j for $1 \leq j \leq m$ are defined as follows:

- Suppose the variable X_i is set to true. Then T_i consists of the edges (a_1^i, b_1^i), (a_{2k}^i, a_{2k+1}^i), for $k = 1$ to 3 and (b_{2t}^i, b_{2t+1}^i), for $t = 1$ to 4.
- Suppose X_i is set to false. Then T_i consists of the edges (a_1^i, a_2^i), (a_3^i, a_4^i), (a_5^i, b_9^i), (a_6^i, a_7^i), (b_{2t-1}^i, b_{2t}^i), for $t = 1$ to 4.
- Suppose clause C_j is satisfied by its first literal. Then T'_j consists of the edges (c_1^j, d_1^j), (c_2^j, c_3^j), (d_{2t}^j, d_{2t+1}^j), for $t = 1$ to 8.
- Suppose clause C_j is satisfied by its second literal. Then T'_j consists of the edges (c_2^j, d_9^j), (c_1^j, c_3^j), (d_{2t-1}^j, d_{2t}^j), for $t = 1$ to 4, (d_{2t}^j, d_{2t+1}^j) for $t = 5$ to 8.
- Suppose clause C_j is satisfied by its third literal. Then T'_j consists of the edges (c_3^j, d_{17}^j), (c_1^j, c_2^j), (d_{2t-1}^j, d_{2t}^j), for $t = 1$ to 8.

The matching M defined above is a perfect matching. We now analyze its unpopularity factor. Let M' be another matching. We can assume that M' is also perfect (this only increases the unpopularity factor of M.) So $M \oplus M'$ is a set of disjoint cycles. We consider each cycle $\rho \in M \oplus M'$ separately. If ρ has no $(1,1)$ edge, then we are done. So let us assume that ρ has some $(1,1)$ edges. Since M is based on a satisfying assignment of ϕ, none of the edges between a vertex in $\{b_t^i\}_{\forall i, \forall t \in \{1,9\}}$ and a vertex in $\{d_t^j\}_{\forall j, \forall t \in \{1,9,17\}}$ can be a $(1,1)$ edge. Therefore, it can be verified that there are only five types of $(1,1)$ edges.

(1) (b_1^i, b_9^i), when the variable X_i is set to true
(2) (b_9^i, b_8^i), when the variable X_i is set to false
(3) (d_1^j, d_{17}^j), when the first literal of C_j is set to true
(4) (d_9^j, d_8^j), when the second literal of C_j is set to true
(5) (d_{17}^j, d_{16}^j), when the third literal of C_j is set to true

Call edges listed in (1), (2) above as *Type 1* edges and call edges listed in (3), (4), (5) above as *Type 2* edges. Two $(1,1)$ edges e_1 and e_2 in ρ are called *neighbors* if there is no other $(1,1)$ edge in ρ between e_1 and e_2. It is also possible that a $(1,1)$ edge is a neighbor of itself (when there is only one $(1,1)$ edge in ρ).

Let $e \in \rho$ be a $(1,1)$ edge.

- Let $e = (b, b')$ be of Type 1. We refer to its neighboring $(1,1)$ edge that is first encountered when we traverse ρ in the direction of $b, b', \ldots$ as e's $\mathcal{B}$-*neighbor* and its other neighbor as e's $\mathcal{A}$-*neighbor*.
- Let $e = (d, d')$ be of Type 2. We refer to its neighboring $(1,1)$ edge that is first encountered when we traverse ρ in the direction of $d, d', \ldots$ as e's $\mathcal{D}$-*neighbor*, and its other neighbor as e's $\mathcal{C}$-*neighbor*.

Observe that by construction, no two $(1,1)$ edges contain vertices from the same gadget. If a $(1,-1)$ edge e' contains only vertices from the same gadget as those in a $(1,1)$ edge e, we say e' is e's *fellow edge*. Clearly a $(1,-1)$ edge can be the fellow edge of at most one $(1,1)$ edge.

The main idea of our proof is that, for each $(1,1)$ edge in ρ, we find a sufficient number of $(1,-1)$ and $(-1,-1)$ edges to "pay" for it. The claim below shows that between every two neighboring $(1,1)$ edges, there is either a $(-1,-1)$ edge or quite a few $(1,-1)$ edges.

Claim 3 *Let $e \in \rho$ be a $(1,1)$ edge. Suppose e is of Type 1. Then*

(1.1) when we traverse from e to its $\mathcal{B}$-neighbor along ρ, either we first encounter three consecutive $(1,-1)$ edges and they are fellow edges of e, or we encounter a $(-1,-1)$ edge.
(1.2) when we traverse from e to its $\mathcal{A}$-neighbor along ρ, either we first encounter two consecutive $(1,-1)$ edges, and the first $(1,-1)$ edge is a fellow edge of e, or we encounter a $(-1,-1)$ edge.

– Suppose e is of Type 2. Then

(2.1) when we traverse from e to its $\mathcal{D}$-neighbor along ρ, either we first encounter three consecutive $(1,-1)$ edges and they are fellow edges of e, or we encounter a $(-1,-1)$ edge.
(2.2) when we traverse from e to its $\mathcal{C}$-neighbor along ρ, either we first encounter two consecutive $(1,-1)$ edges, or we encounter a $(-1,-1)$ edge. In the former case, the first $(1,-1)$ is either a fellow edge of e or is not a fellow edge of any $(1,1)$ edge.

We first finish the proof of Lemma 9 using the above claim and then prove the claim. If there is a $(-1,-1)$ edge between two neighboring $(1,1)$ edges, then we assign a -1 to each $(1,1)$ edge. If there is no $(-1,-1)$ edge between two neighboring $(1,1)$ edges e_1 and e_2, then we distribute the $(1,-1)$ edges among them as follows.

- If e_2 is e_1's $\mathcal{B}$- or $\mathcal{D}$-neighbor, then by (1.1) and (2.1) of the above claim, there are three $(1,-1)$ fellow edges of e_1. We assign them to e_1.
- If e_2 is e_1's $\mathcal{A}$-neighbor, then by (1.2) of the above claim, there are at least 2 $(1,-1)$ edges between them and one of them (the one that is closer to e_1) is a fellow edge of e_1. We assign this edge to e_1.
- If e_2 is e_1's $\mathcal{C}$-neighbor, then by (2.2) of the above claim, there are at least 2 $(1,-1)$ edges between them. Assign the one that is closer to e_1 to e_1.

The critical thing is that *we never assign a $(1,-1)$ edge to multiple $(1,1)$ edges in ρ*. In all but one case, we assign a $(1,-1)$ edge e' to a $(1,1)$ edge e only when the former is a fellow edge of the latter. The one subtle case is that when e_1 is of Type 2, we assign some $(1,-1)$ edge e^* between it and its $\mathcal{C}$-neighbor e_2 to e_1, and e^* may *not* be a fellow edge of e_1. But note that (2.2) of the claim states that e^* cannot be a fellow edge of any other $(1,1)$ edge also.

We now show that it cannot happen that e^* is also assigned to e_2. First note that if e_1 is e_2's $\mathcal{A}$-, $\mathcal{B}$-, or $\mathcal{D}$-neighbor, we only assign the fellow $(1,-1)$ edges of e_2 to e_2. So none of these edges can be e^*. If e_1

happens to be e_2's $\mathcal{C}$-neighbor, observe that there are at least two $(1,-1)$ edges between e_1 and e_2, and by our procedure, we will assign the $(1,-1)$ edge closer to e_1 to e_1 and the one closer to e_2 to e_2. Thus there is no danger that we assign e^* to both.

We now prove that for each $(1,1)$ edge e, the number of 1's and -1's we assign to it guarantees that the ratio of 1's against -1's is bounded by 1.5. This will complete the proof. (Those $(1,-1)$ and $(-1,-1)$ edges that are not assigned to any $(1,1)$ edge in the above procedure obviously have the ratio of at most 1 among themselves).

- Suppose that there is a $(-1,-1)$ edge between e and both of its neighbors, then we have 2 1's and 2 -1's.
- Suppose that there is a $(-1,-1)$ edge between e and one of its neighbors but there is none between e and its other neighbor e'. Consider the following sub-cases.
  - If e' is the $\mathcal{B}$- or $\mathcal{D}$- neighbor of e, then we give three $(1,-1)$ edges to e. In total, we have 5 1's and 4 -1's.
  - If e' is the $\mathcal{A}$- or $\mathcal{C}$- neighbor of e, then we give one $(1,-1)$ edge to e. In total, we have 3 1's and 2 -1's.
- Suppose that there is no $(-1,-1)$ edge between e and both of its neighbors. Then we assign to e three $(1,-1)$ edges between e and its $\mathcal{B}$- or $\mathcal{D}$-neighbor, and one extra $(1,-1)$ edge between e and its $\mathcal{A}$- or $\mathcal{C}$-neighbor. In total, we have 6 1's and 4 -1's. This finishes the proof of the lemma. □

Proof of Claim 3: To show (1.1) and (1.2) for Type 1 edges e, we assume that $e = (b_1^i, b_9^i)$. The other case (i.e., $e = (b_9^i, b_8^i)$) follows symmetrically. As $M(b_9^i) = b_8^i$, if b_8^i is matched to a vertex in the "$\cdots$" part of its list in M', then we have a $(-1,-1)$ edge, since by our construction of M, all vertices prefer their partners to those in the "$\cdots$" of their preference lists. So the only other possibility is that $M'(b_8^i) = b_7^i$ and this gives us a $(1,-1)$ edge and it is a fellow edge of e. Now repeating the same argument on $M(b_7^i) = b_6^i$, and on $M(b_5^i) = b_4^i$ gives the proof of (1.1).

For (1.2), first note that $M(b_1^i) = a_1^i$. If $M'(a_1^i)$ is any vertex belonging to the "$\cdots$" part of the preference list of a_1^i, then we have a $(-1,-1)$ edge. So there are only two other possibilities to consider: either $M'(a_1^i) = a_2^i$, or $M'(a_1^i) = a_7^i$ and both would give us a $(1,-1)$ edge, which is a fellow edge of e.

If $M'(a_1^i) = a_2^i$, then $M(a_2^i) = a_3^i$. In M', independent of whether a_3^i is matched to a_4^i or a vertex in $\pi(a_3^i)$, we have another $(1,-1)$ edge.[3] This is because a_4^i prefers a_5^i to a_3^i, and all vertices in $\{c_t^j\}_{\forall j, \forall 1 \le t \le 3}$ (these are contained in $\pi(a_3^i)$) prefer their partners in M to any vertex listed in $\pi(c_t^j)$. The other case, i.e., $M'(a_1^i) = a_7^i$ can be argued similarly and this completes the proof of (1.2).

To show (2.1) and (2.2) for Type 2 edges e, we assume $e = (d_1^j, d_{17}^i)$. The other cases follow symmetrically. (2.1) can be proved using essentially the same argument as was used in proving (1.1). For (2.2), consider the following cases.

- Suppose $M'(c_1^j) = c_2^j$ or c_3^j. Then we have a $(1,-1)$ edge, which is a fellow edge of e. If $M'(c_1^j) = c_2^j$, then $M(c_2^j) = c_3^j$, and c_3^j can be matched in M' to either d_{17}^j, or to vertices in $\pi(c_3^j)$, or to those in the "$\cdots$" part of its list. In all cases, we have another $(1,-1)$ edge or a $(-1,-1)$ edge. The case when $M'(c_1^j) = c_3^j$ follows symmetrically.
- Suppose $M'(c_1^j) \in \pi(c_1^j)$. Then $M'(c_1^j) = a_3^i$ or $M'(c_1^j) = a_7^i$ for some variable X_i. Suppose that $M'(c_1^j) = a_3^i$. If X_i is set to false in the truth assignment, then $M(a_3^i) = a_4^i$, and as a_3^i prefers a_4^i to c_1^j, we have a $(-1,-1)$ edge. So assume that X_i is set to true and $M(a_3^i) = a_2^i$. Then we have a $(1,-1)$ edge consisting of c_1^j and a_3^i. Moreover, such an edge cannot be a fellow edge of any $(1,1)$ edge. Now as $M(a_3^i) = a_2^i$, if a_2^i is matched to a_1^i in M', then we have yet another $(1,-1)$ edge and if a_2^i is matched to some other vertex in the "$\cdots$" part of its list, then we again have a $(-1,-1)$ edge. This completes the proof of (2.2). The case that $M'(c_1^j) = a_7^i$ follows a symmetrical argument to the one above. This finishes the proof of the claim. □

Conclusions and Open problems. We considered the problem of computing a least unpopularity matching in a roommates instance $G = (V, E)$ with strict preferences and incomplete lists. We showed that this problem is NP-hard. It is, in fact, NP-hard to compute a matching whose unpopularity factor is at most

[3] Note that this edge may not be a fellow edge of e, but this does not affect the statement of the claim.

$4/3 - \epsilon$ of the optimal, for any $\epsilon > 0$. On the positive side, we showed that there always exists a matching whose unpopularity factor is $O(\log |V|)$ and such a matching can be computed in linear time.

Our positive result is also an $O(\log |V|)$ approximation algorithm for the problem of computing a least unpopularity factor matching. An open problem is to design an algorithm with a better approximation guarantee. Another open problem is to design an efficient algorithm to determine if G admits a popular matching or not. No polynomial time algorithm is known for this problem.

References

1. D.J. Abraham, R.W. Irving, T. Kavitha, and K. Mehlhorn. Popular matchings. *SIAM Journal on Computing*, 37(4): 1030–1045, 2007.
2. P. Biro, R. W. Irving, and D. F. Manlove. Popular matchings in the Marriage and Roommates problems In *Proceedings of CIAC 2010*, pages 97–108.
3. D. Gale and L.S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69: 9–15, 1962.
4. P. Gärdenfors. Match making: assignments based on bilateral preferences. *Behavioural Sciences*, 20: 166–173, 1975.
5. M. Garey and D. Johnson. *Computers and Intractablility.* Freeman, 1979.
6. C.-C. Huang and T. Kavitha. Popular Matchings in the Stable Marriage Problem. *Proceedings of ICALP 2011*, pp. 666-677.
7. R.W. Irving. An Efficient Algorithm for the "Stable Roommates" Problem. *Journal of Algorithms*, 6: 577–595, 1985.
8. T. Kavitha. Popularity vs Maximum cardinality in the stable marriage setting. *Proceedings of SODA 2012* (to appear).
9. T. Kavitha, J. Mestre, and M. Nasre. Popular mixed matchings. In *Proceedings of ICALP 2009*, pp. 574–584.
10. M. Mahdian. Random popular matchings. In *Proceedings of the ACM EC 2006*, pp. 238–242.
11. D.F. Manlove and C. Sng. Popular matchings in the capacitated house allocation problem. In *Proceedings of ESA 2006*, pp. 492–503.
12. M. McCutchen. The least-unpopularity-factor and least-unpopularity-margin criteria for matching problems with one-sided preferences. In *Proceedings of LATIN 2008*, pp. 593–604.
13. J. Mestre. Weighted popular matchings. In *Proceedings of ICALP 2006*, pp. 715–726.
14. J. J. M. Tan. A necessary and sufficient condition for the existence of a complete stable matching. *Journal of Algorithms*, 12: 154–178, 1991.

Appendix

An instance where every matching has large unpopularity factor

We show a graph $G = (V, E)$ on $N = 3^k$ vertices where $u(M) \geq 2k$ for all matchings M. Let $V = \{a_1, \ldots, a_{N/3}, b_1, \ldots, b_{N/3}, c_1, \ldots, c_{N/3}\}$ where every $a \in A = \{a_1, \ldots, a_{N/3}\}$ has its own permutation of vertices in $A \setminus \{a\}$ as its top $N/3 - 1$ neighbors, followed by $b_1, \ldots, b_{N/3}$ (in this order), further followed by $c_1, \ldots, c_{N/3}$ (in this order). Symmetrically, every $b \in B = \{b_1, \ldots, b_{N/3}\}$ has its own permutation of vertices in $B \setminus \{b\}$ as its top $N/3 - 1$ neighbors, followed by $c_1, \ldots, c_{N/3}$ (in this order), further followed by $a_1, \ldots, a_{N/3}$ (in this order). Similarly, every $c \in C = \{c_1, \ldots, c_{N/3}\}$ has its own permutation of vertices in $C \setminus \{c\}$ as its top $N/3 - 1$ neighbors, followed by $a_1, \ldots, a_{N/3}$ (in this order), further followed by $b_1, \ldots, b_{N/3}$ (in this order). Recursively, A can be further partitioned to 3 subsets A_0, A_1, A_2, where for $i = 0, 1, 2$, every $a \in A_i$ has its own permutation of vertices in $A_i \setminus \{a\}$ as its top $N/9 - 1$ neighbors, followed by vertices in $A_{(i+1) \bmod 3}$ in a fixed order, further followed by vertices in $A_{(i+2) \bmod 3}$ in a fixed order and this goes on. Similarly, this holds for B and C also.

Lemma 10. *Any matching M in G satisfies $u(M) \geq 2k$, where $k = \log_3 N$.*

Proof. Let M be any matching in G. We can assume that M leaves just 1 vertex unmatched, otherwise its unpopularity factor would be ∞. We know that the vertex set of G can be partitioned into 3 subsets A, B, and C. Assume without loss of generality that the vertex unmatched by M is in C. Since the sizes of A, B, and C are odd, each of A, B, and C has an *odd* number of vertices that are not matched to one of

their own kind. That is, there are an odd number of vertices in A that are not matched by M to a partner in A; there are an odd number of vertices in B that are not matched by M to a partner in B; and there are an odd number of vertices in C that are not matched by M to a partner in C (and this includes the unmatched vertex of C).

Thus there are an even number of vertices in C that are not matched among themselves, but are matched to vertices in $A \cup B$. Recall that our job is to come up with a matching M' such that $\Delta(M, M') \geq 2k$. We define M' as follows:

- M' matches all the vertices in B (similarly, C) that are matched by M to another vertex in B (resp., C) in the same way as M.
- The vertices in B that are matched by M to vertices in C remain matched by M' in the same way as in M.
- Now there are 2 cases now to consider: whether there are an *even* number or an *odd* number of vertices in B that are matched to vertices in A.
  - In the former case, there are an odd number of vertices of C that are matched to vertices in A: this odd number of vertices of C along with the unmatched vertex in M form an even sized set - we will match these vertices among themselves arbitrarily. Similarly, the even number of vertices in B that are matched by M to vertices in A - we will match these vertices of B arbitrarily among themselves in M'.
  - In the latter case, there are an even number of vertices of C that are matched to vertices in A - match all these vertices among themselves arbitrarily. There have to be an odd number of vertices of B that are matched to vertices in A - match exactly one of these to the unmatched vertex in C and match the remaining among themselves arbitrarily.

Note that all the vertices in $B \cup C$ whose assignments in M' are different from M, are all happier with M' than M, and this includes at least 1 vertex from B and at least 1 vertex from C. Also, note that every vertex in $B \cup C$ is matched in M' and no vertex in $B \cup C$ is matched to a vertex in A. We would now like to extend M' to A also. For that purpose, we focus exclusively on how M matches vertices in A.

So instead of the original matching M that matched some vertices of A to vertices in $B \cup C$ which is not possible in M' now, we will look at a slightly modified matching M where all the vertices of A that were matched by M to another vertex in A remain matched the same way, however all the vertices of A that were matched by M to vertices in $B \cup C$ get rematched among themselves arbitrarily. Since every vertex in $B \cup C$ is matched in M', this will leave us with exactly 1 vertex of A that is unmatched now.

Now consider this revised matching M on A. By applying the above strategy (whatever we did on A, B, C to get M' on $B \cup C$) on A_0, A_1, A_2 and recursively continuing till we are at a set of size 1, in which case this lone vertex is left unmatched, we come up with the matching M' on A.

We have gone through k levels of recursion here and in each level of recursion, we fix partners in M' for 2/3 fraction of the current set of vertices (for instance, for vertices in $B \cup C$ in our first level of recursion) and every vertex in this 2/3 fraction that gets a new assignment is better off in M' than its assignment in M. Observe that there are at least 2 vertices that get new partners in each level of recursion. Thus there are at least $2k$ vertices that are happier with M' than with M. Exactly one vertex is unhappier with M' than with M - this is the vertex left unmatched in M'. Hence $\Delta(M, M') \geq 2k$, thus $u(M) \geq 2k$. □

A bad instance for our algorithm

We construct an instance $G = (V, E)$ on $N = 3^k$ vertices where our algorithm, as presented at the beginning of Section 2.1, returns a matching M with $u(M) \geq 2k$ while there exists a matching M' with $u(M') = 2$. This shows that this algorithm has an approximation factor that is at least $k = \log_3 |V|$. Note that in the original algorithm, all the odd indexed vertices in the ordered sets form the set X_i (recall that the vertices in X_i propose to those in $V_i \setminus X_i$). However in our implementation (refer to Section 2.1), we choose the vertices that constitute X_i in a more careful manner. Nevertheless, we will show later that with some slight modification of the instance, we still can guarantee that the approximation ratio of our final algorithm is $\Theta(\log |V|)$.

Let the vertices be $a_0, \cdots, a_{N-1}$. We now describe our instance in the form of *layers*: all the vertices $a_0, \cdots, a_{N-1}$ belong to the first layer; all the vertices in $\cup_{i=0}^{N/3-1}\{a_{3i}\}$ also belong to the second layer; and

more generally, the vertices in $\cup_{i=0}^{N/3^t-1}\{a_{3^{t-1}i}\}$ also belong to the t-th layer. So $a_{3^p i}$ where $p \neq 0 \pmod 3$ belongs to the first $p+1$ layers and no more.

Roughly speaking, we will construct the preferences in such a way that when we apply our algorithm, if a_u is matched to a_v, then both of them belong only to the first t layers for some t. We describe the preference lists of all vertices in the first layer. For all $0 \leq i \leq \frac{N}{3}-1$, let the preference lists be as follows:

$$a_{3i} : a_{3i+1}\ a_{3i+2}\ \cdots; \qquad a_{3i+1} : a_{3i+2}\ a_{3i}; \qquad a_{3i+2} : a_{3i}\ a_{3i+1}.$$

Note that both a_{3i+1} and a_{3i+2} belong only to the first layer, while a_{3i} belongs to more layers. So a_{3i} has a longer preference list (the $\cdots$ part) and the $\cdots$ part will be explained below. For the second layer, for all $0 \leq i \leq \frac{N}{9}-1$, let the preference lists be as follows:

$$a_{9i} : a_{9i+1}\ a_{9i+2}\ a_{9i+3}\ a_{9i+6} \cdots; \qquad a_{9i+3} : a_{9i+4}\ a_{9i+5}\ a_{9i+6}\ a_{9i}; \qquad a_{9i+6} : a_{9i+7}\ a_{9i+8}\ a_{9i}\ a_{9i+3}.$$

Again, observe that a_{9i+3} and a_{9i+6} belong to only the first two layers, while a_{9i} belongs to more layers and it has a longer preference list. We construct the rest of the preference lists recursively. For the t-th layer, for any $0 \leq i \leq \frac{N}{3^t}-1$, let the preference lists be as follows:

$$a_{3^t i} : \cdots\ a_{3^t i+3^{t-1}}\ a_{3^t i+2*3^{t-1}} \cdots; \qquad a_{3^t i+3^{t-1}} : \cdots\ a_{3^t i+2*3^{t-1}}\ a_{3^t i};$$
$$a_{3^t i+2*3^{t-1}} : \cdots\ a_{3^t i}\ a_{3^t i+3^{t-1}}.$$

The initial $\cdots$ part of the three preference lists refers to the part of the preferences that have been built in earlier recursions. The second $\cdots$ part of $a_{3^t i}$ refers to the preference that will be built in the next iteration. In the special case, $t = k-1$ (the last recursion), this second $\cdots$ part of $a_{N/3}$ is empty. This completes our construction. See Table 3 for an example of $N = 3^3 = 27$ vertices.

Table 3. The preference lists of 27 vertices in $G = (V, E)$.

Vertex	Preference List	Vertex	Preference List	Vertex	Preference List
a_0	$a_1\ a_2\ a_3\ a_6\ a_9\ a_{18}$	a_9	$a_{10}\ a_{11}\ a_{12}\ a_{15}\ a_{18}\ a_0$	a_{18}	$a_{19}\ a_{20}\ a_{22}\ a_{24}\ a_0\ a_9$
a_1	$a_2\ a_0$	a_{10}	$a_{11}\ a_9$	a_{19}	$a_{20}\ a_{18}$
a_2	$a_0\ a_1$	a_{11}	$a_9\ a_{10}$	a_{20}	$a_{18}\ a_{19}$
a_3	$a_4\ a_5\ a_6\ a_0$	a_{12}	$a_{13}\ a_{14}\ a_{15}\ a_9$	a_{21}	$a_{22}\ a_{23}\ a_{24}\ a_{18}$
a_4	$a_5\ a_3$	a_{13}	$a_{14}\ a_{12}$	a_{22}	$a_{23}\ a_{21}$
a_5	$a_3\ a_4$	a_{14}	$a_{12}\ a_{13}$	a_{23}	$a_{21}\ a_{22}$
a_6	$a_7\ a_8\ a_0\ a_3$	a_{15}	$a_{16}\ a_{17}\ a_9\ a_{12}$	a_{24}	$a_{25}a_{26}a_{18}a_{21}$
a_7	$a_8\ a_6$	a_{16}	$a_{17}\ a_{15}$	a_{25}	$a_{26}\ a_{24}$
a_8	$a_6\ a_7$	a_{17}	$a_{15}\ a_{16}$	a_{26}	$a_{24}\ a_{25}$

Apply our algorithm on the constructed instance.

- In the first iteration, $\cup_{i=0}^{N/3-1}\{\langle a_{3i}, a_{3i+1}, a_{3i+2}\rangle\}$ is a stable partition. The set X_0 is $\cup_{i=0}^{N/3-1}\{a_{3i+1}\}$. The Gale-Shapley algorithm pairs a_{3i+1} with a_{3i+2} while leaving a_{3i} alone for $0 \leq i \leq \frac{N}{3}-1$.
- In the second iteration, $\cup_{i=0}^{N/9-1}\{\langle a_{9i}, a_{9i+3}, a_{9i+6}\rangle\}$ is a stable partition. The set X_1 is $\cup_{i=0}^{N/9-1}\{a_{9i+3}\}$. The Gale-Shapley algorithm pairs a_{9i+3} with a_{9i+6} while leaving a_{9i} alone for $0 \leq i \leq \frac{N}{9}-1$.
- Repeating this process, the final matching is

$$M = \cup_{i=0}^{N/3-1}\{(a_{3i+1}, a_{3i+2})\} \cup_{i=0}^{N/9-1}\{(a_{9i+3}, a_{9i+6})\} \cup \cdots \cup_{i=0}^{N/3^t-1}\{(a_{3^t i+3^{t-1}}, a_{3^t i+2*3^{t-1}})\} \cup \cdots \cup \{(a_{N/3}, a_{2N/3})\},$$

with the vertex a_0 left unmatched.

Consider the following alternating path ρ:

$$a_0 - a_{2N/3} - a_{N/3} - a_{N/3+2N/9} - a_{N/3+N/9} - \cdots -$$
$$a_{(\sum_{i=1}^{t-1} N/3^i)+2N/3^t} - a_{(\sum_{i=1}^{t-1} N/3^i)+N/3^t} - \cdots - a_{(\sum_{i=1}^{k-1} N/3^i)+2} - a_{(\sum_{i=1}^{k-1} N/3^i)+1},$$

where a_0 is the only vertex unmatched in M. Observe that a_0, $a_{2N/3}$, $a_{N/3}$ all belong to k layers and by the preferences we construct, the edge $(a_0, a_{2N/3})$ is a $(1,1)$ edge. Similarly, $a_{N/3}$, $a_{N/3+2N/9}$, $a_{N/3+N/9}$ all belong to the first $(k-1)$ layers and by the preferences, the edge $(a_{N/3}, a_{N/3+2N/9})$ is a $(1,1)$ edge, and so on. Therefore, every non-matched edge in ρ is a $(1,1)$ edge. Since there are in total k such edges and only one vertex, that is $a_{(\sum_{i=1}^{k-1} N/3^i)+1}$, which remains unmatched in $M \oplus \rho$, we have $\Delta(M, M \oplus \rho) \geq 2k$, thus $u(M) \geq 2k$.

We now construct a matching M'. Let $M' = \{(a_{3i}, a_{3i+1})\}_{i=0}^{N/3-1}$, with all the vertices a_{3i+2}, for $i = 0, \ldots, \frac{N}{3} - 1$, left unmatched. We claim that $u(M') = 2$.

To see this, note that for all $0 \leq i \leq \frac{N}{3} - 1$, (a_{3i+1}, a_{3i+2}) is a $(1,1)$ edge and (a_{3i}, a_{3i+2}) is a $(1,-1)$ edge. Moreover, all other possible edges not in M' are $(-1,-1)$ edges. By this observation, the alternating/augmenting paths that have the largest ratio of 1's against -1's are a_{3i+2}-a_{3i+1}-a_{3i}-a_{3j}-a_{3j+1}-a_{3j+2} and a_{3i+2}-a_{3i+1}-a_{3i}, the former has four 1's and two -1's while the latter has two 1's and one -1. So we conclude that $u(M') = 2$.

We now consider our refined algorithm that takes $O(n+m)$ time to find the matching. In our current constructed instance, in the first iteration, $\langle a_0, a_1, a_2 \rangle$ is a tuple in the stable partition. Instead of choosing the odd indexed vertex, which is a_1, the refined algorithm chooses the vertex with at least $m_j/3$ incident edges, where m_j is the total number of edges incident on $a_0 \cup a_1 \cup a_2$. This is the vertex a_0. Thus our refined algorithm chooses $\{a_{3i}\}_{i=0}^{N/3-1}$ as the set X_0 to do the proposing. Thus the outcome will be M', instead of M.

We modify our instance to "fool" our refined algorithm so that it would behave the same as the original algorithm, returning M. Introduce edges between vertices of the set $\cup_{i=0}^{N/3-1}\{a_{3i+1}\}$. For each vertex a_{3i+1}, $0 \leq i \leq N/3 - 1$, the ranks on these newly added edges are arbitrary, but all these ranks are worse than those on the existing edges (a_{3i+1}, a_{3i+2}) and (a_{3i+1}, a_{3i}). Similarly, for vertices in the t-th layer, where $1 < t < k$, let the vertices in $\cup_{i=0}^{N/3^t-1}\{a_{3^t i+3^{t-1}}\}$ have edges with one another and let the ranks on these edges be worse than the old edges incident on these vertices.

We now consider the first iteration of the refined algorithm. Clearly, the stable partition remains unchanged after we add all those extra edges. For each part $\cup_{i=0}^{N/3-1}\{\langle a_{3i}, a_{3i+1}, a_{3i+2} \rangle\}$, a_{3i+1} has $2 + (N/3 - 1) = 1 + N/3$ incident edges, a_{3i+2} has only 2 edges, while a_{3i} has at most $4 + N/9 - 1 = 3 + N/9$ edges. (This happens when a_{3i} belongs only to the first two layers; if a_{3i} belongs to only the first t layers, it has $2k + N/3^t - 1$ incident edges, which is even less). As a result, $\cup_{i=0}^{N/3-1}\{a_{3i+1}\}$ becomes the set X_0. The argument for the following iterations is essentially the same. So the matching M with $u(M) \geq 2k$ will be output by our refined algorithm. Finally, the matching M' still obeys $u(M') = 2$, since all these extra edges are $(-1,-1)$ edges with respect to M'.