Egerváry Research Group
on Combinatorial Optimization

Technical reportS

TR-2021-10. Published by the Egerváry Research Group, Pázmány P. sétány 1/C, H–1117, Budapest, Hungary. Web site: www.cs.elte.hu/egres. ISSN 1587–4451.

Polynomially tractable cases in the popular roommates problem

Erika Bérczi-Kovács, Ágnes Cseh, Kata Kosztolányi, and Attila Edmund Mályusz

---

October 2021

Polynomially tractable cases in the popular roommates problem

Erika Bérczi-Kovács, Ágnes Cseh, Kata Kosztolányi, and Attila Edmund Mályusz

Abstract

The input of the popular roommates problem consists of a graph $G = (V, E)$ and for each vertex $v \in V$, strict preferences over the neighbors of v. Matching M is more popular than M' if the number of vertices preferring M to M' is larger than the number of vertices preferring M' to M. A matching M is called *popular* if there is no matching M' that is more popular than M.

Only recently Faenza et al. [12] and Gupta et al. [15] resolved the long-standing open question on the complexity of deciding whether a popular matching exists in a popular roommates instance and showed that the problem is NP-complete. In this paper we identify a class of instances that admit a polynomial-time algorithm for the problem. We also test these theoretical findings on randomly generated instances to determine the existence probability of a popular matching in them.

1 Introduction

Our input is an instance $G = (V, E)$ of the stable roommates problem with strict and possibly incomplete preference lists. A matching M is *stable* if there is no *blocking pair* with respect to M, in other words, there is no pair of vertices (a, b) such that a is either unmatched or prefers b to $M(a)$ (a's partner in M) and similarly, b is either unmatched or prefers a to $M(b)$. Irving [22] gave a polynomial-time algorithm to decide whether G admits a stable matching.

In this paper we consider *popularity*, a notion that is more relaxed than stability. For a vertex $v \in V$, v's ranking over its neighbors can be extended naturally to a ranking over matchings as follows: v prefers matching M to matching M' if (i) v is matched in M and unmatched in M' or (ii) v is matched in both and prefers $M(v)$ to $M'(v)$.

The motivation of popularity comes from voting. Suppose an election is held between M and M' where each vertex casts a vote for the matching it prefers. We call a matching M popular in the instance if it never loses an election to another matching M'. In voting terminology, each popular matching is a weak *Condorcet winner* [4] in the corresponding voting instance.

Stable matchings are always popular [3]. Thus, if an instance admits a stable matching, then the existence of a popular matching is also guaranteed, and it can be found using Irving's algorithm for finding a stable matching [22]. Some instances of the stable roommates problem do not admit a stable solution, yet they admit a popular matching, as demonstrated by Figure 1, first presented by Biró et al. [2].

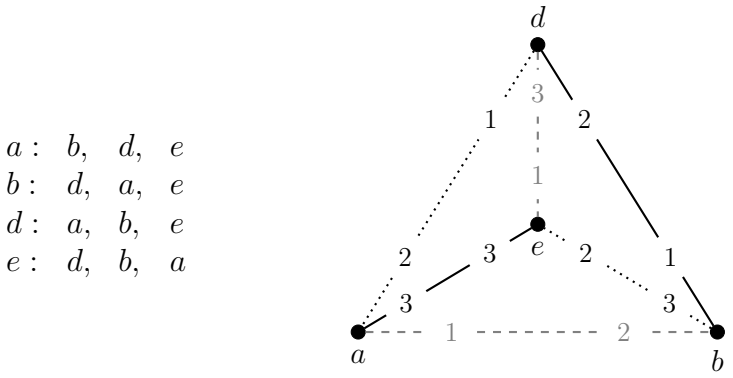

Figure 1: The preferences of each vertex are shown on the left hand-size and also as labels on the edges in the graph. The dashed gray edges mark the popular matching $M = \{(a, b), (d, e)\}$. It is blocked by the edge (b, d). The instance admits no stable matching.

Validating whether a given matching is popular can be done in polynomial time [2]. Only recently Faenza et al. [12] and Gupta et al. [15] resolved the long-standing [2, 5, 18, 20, 29] open problem on the complexity of deciding whether a popular matching exists in a popular roommates instance and showed that the problem is NP-complete. This hardness extends to graphs with complete preference lists [7]. However, this latter hardness result is only valid for complete graphs with an *even* number of vertices. It is known that when n, the number of vertices in G is *odd*, a matching in a complete graph G on n vertices is popular only if it is stable [7]. Since Irving's algorithm [22] can decide if G admits a stable matching or not, the popular roommates problem in a complete graph G can be efficiently solved for odd n. It is easy to see that the proof in [7] extends to graphs with minimum degree $n - 2$ as well: even in those instances, a matching is popular if and only if it is stable.

1.1 Our contribution

In this paper, we stretch this observation even further and show that for *high-degree* graphs with an *odd* n, there is a polynomial algorithm that solves the popular roommates problem. More precisely, our algorithm runs in polynomial time for a constant c, where $n - c$ is the minimum degree in the graph.

Our key technical result is an algorithm to decide whether there exists a popular matching that leaves exactly a given non-empty set of vertices uncovered. We prove that popular matchings with this property can be decomposed into a stable and a popular part. We iterate through the possible popular parts and construct the fitting stable part in a carefully designed subroutine.

We also test our algorithm on randomly generated Erdős–Rényi graphs and conclude that on graphs with a high minimum degree, our method is significantly faster than checking the matchings in the instance for popularity. Furthermore, we also tally the instances that admit a popular matching, and within this set, the instances that admit no stable matching. Ours is the first experimental study on the existence probability of popular matchings in randomly generated non-bipartite graphs.

1.2 Literature review

The notion of popularity was first introduced for bipartite graphs in 1975 by Gärdenfors—popular matchings always exist in bipartite graphs since stable matchings always exist here [13] and every stable matching is popular [14]. The proof that every stable matching is popular holds in non-bipartite graphs as well [3]; in fact, it is easy to show that every stable matching is a min-size popular matching in both settings [17].

Huang and Kavitha [20] showed that the polytope of popular fractional matchings is half-integral in the non-bipartite case. This means that one can compute a maximum weight popular half-integral matching in polynomial time. They also showed that the problem of computing an integral maximum weight popular matching in a non-bipartite instance is NP-hard. Finding a maximum cardinality popular matching in instances admitting a popular matching has also been shown to be NP-hard [24]. The breakthrough regarding the complexity of the popular roommates problem came after these works: two independent NP-completeness proofs, by Faenza et al. [12] and Gupta et al. [15], were published in 2019.

Most positive results in the setting are also very recent. The only known tractable subclasses of popular matchings are the class of stable matchings and the class of so-called 'strongly dominant matchings', which is a subclass of max-size popular matchings [12]. If the underlying graph G has a bounded treewidth, then the min-cost popular matching problem can be solved in polynomial time [12]. Kavitha provided a simply exponential time algorithm for the popular roommates problem [25]. Interestingly, its running time was slightly decreased very recently by Palmer and Pálvölgyi [31], who improved the upper bound on the number of stable matchings in bipartite matching instances. However, her algorithm stays exponential for the special class of graphs we discuss in this paper. Huang and Kavitha [18] proved that each roommates instance admits a matching that has the approximability measure called 'unpopularity factor' of $O(\log n)$.

The existence probability of a stable matching in randomly generated roommates instances has been an actively researched topic for decades, which lead to both theoretical [22, 32, 33] and experimental findings [34, 30, 10]. Until now, randomly generated popular matching instances have only been studied in the context of bipartite graphs, both with one-sided [28, 23, 35] and two-sided preferences [1].

2 Preliminaries

Consider a graph $G = (V, E)$ with n vertices and m edges. Let $N(u)$ denote the vertices adjacent to $u \in V$ in G, and let $N(U)$ for $U \subseteq V$ be the set of vertices that are adjacent to at least one vertex in U. Furthermore, let $G[U]$ be the graph spanned by the vertex set U.

Each vertex ranks all adjacent vertices in a strict order of preference. We write $v \succ_u w$ if u prefers v to w. A *matching* $M \subseteq E$ is a set of edges such that each vertex is incident to at most one edge in M. For convenience, we denote u's partner in M by $M(u)$, and write $M(u) = u$ if u is unmatched in M. For a set $X \subseteq V$, $M(X) = \bigcup_{v \in X} M(v)$. We assume that $v \succ_u u$ for all $v \in N(u)$. In other words, each vertex prefers being matched along any of its edges to staying unmatched.

We now introduce the edge labeling technique that is standard in the field [17]. Let M be a matching in G. For each edge $(u, v) \notin M$, we define the vote of vertex u between the edge (u, v) and M as follows:

$$\mathsf{vote}_u(v, M) = \begin{cases} + & \text{if } v \succ_u M(u); \\ - & \text{if } M(u) \succ_u v. \end{cases}$$

The next step is to label every edge $(u, v) \notin M$ by the pair $(\mathsf{vote}_u(v, M), \mathsf{vote}_v(u, M))$. Thus every non-matching edge has a label in $\{(\pm, \pm)\}$. Note that an edge is labeled $(+, +)$ if and only if it blocks M. Let G_M be the subgraph of G obtained by deleting edges labeled $(-, -)$ from G. The following theorem characterizes popular matchings in G.

Theorem 2.1 (Huang and Kavitha [17])**.** *M is popular in G if and only if G_M does not contain any of the following with respect to M:*

1. *an alternating cycle with a $(+, +)$ edge;*
2. *an alternating path with two disjoint $(+, +)$ edges;*
3. *an alternating path with a $(+, +)$ edge and an unmatched vertex as an end vertex.*

Using the above characterization, it can be easily checked whether a given matching M is popular [26]: M is popular if and only if it is a maximum weight perfect matching in the weighted graph we get by first adding a loop to each vertex and then assigning weight -1 to those loops that belong to vertices covered by M, 2 to all $(+, +)$ edges, -2 to all $(-, -)$ edges, and 0 to all other edges and loops. However, this only settles the complexity of verification.

Structure of the paper. We first explain the high-level idea of our algorithm in Section 3. Its proof of correctness and the exact implementation details are then given in Section 4. Section 5 contains a detailed example of the execution. In Section 6 we analyze the performance of our algorithm on randomly generated graphs. We pose open questions in Section 7. Two full example runs are to be found in the appendix.

3 Our algorithm

As we have already observed in Section 1, if the instance admits a stable matching, then it is popular as well, so the existence question is only interesting for instances that do not admit any stable matching. From now on, we restrict our attention to these instances.

We are given an instance of the popular roommates problem that does not admit a stable matching. Let U be a non-empty set of vertices. We now present an algorithm with which it can be checked whether there is a popular matching in G that leaves exactly U uncovered. Notice that the answer is trivially 'NO' if U spans any edge in G.

To a given U, let $Z = V \setminus N(U) \setminus U$ denote the set of those vertices that are not in U and also not adjacent to *any* vertex in U. It is clear that the set Z is well-defined for each set U. In our algorithm (see Algorithm 1 for a pseudocode), we build a set $\mathcal{P}_Z$ of initial matchings and test for each $P_Z \in \mathcal{P}_Z$ whether it can be completed to a popular matching that leaves exactly U uncovered. The construction of $\mathcal{P}_Z$ is simple: we list all matchings in which each edge has at least one end vertex in Z and also cover each vertex in Z. Then, we examine each such $P_Z \in \mathcal{P}_Z$ in three steps.

1. If P_Z is not popular in $G' = G[Z \cup P_Z(Z) \cup U]$, then we stop testing this P_Z and conclude that P_Z cannot be extended to a desired popular matching.

2. If there is no path $v - P_Z(v) - P_Z(w) - w$ with $(P_Z(v), P_Z(w))$ blocking P_Z, then we output the same message and stop examining P_Z.

3. Otherwise, there is at least one alternating path traversing through a blocking edge in G'. In Claim 3 we show that matching P_Z can be extended to a popular matching in G with edge set S leaving exactly U uncovered if and only if S is a stable complete matching in a transformed graph. Later, in Section 4.4 we will show how the existence of a desired stable matching can be checked in polynomial time.

4 Proof of correctness

In Sections 4.1-4.3 we prove the necessity of our three tests in lines 4-6 in our pseudocode. Then in Section 4.4 we elaborate on the implementation of the last one of these. Finally in Section 4.5 we complete the proof of correctness and provide a running time analysis.

4.1 First test

Our first claim shows the necessity of the first step in line 4. More precisely, we show that if a matching P_Z is not popular in G', then it cannot be extended to a popular matching in G, and thus we can proceed to testing the next candidate for P_Z.

Claim 1. If the constructed matching P_Z is not popular in G', then no matching P containing P_Z is popular in G.

Algorithm 1 Routine checking if set $U \neq \varnothing$ can be the set of uncovered vertices in a popular matching

Input: A popular roommates problem instance that admits no stable matching and a non-empty set of vertices $U \subseteq V$.

Output:A popular matching covering exactly $V \setminus U$ or a 'NO' answer.

1: $Z :=$ set of vertices in $V \setminus U$ that are not adjacent to any vertex in U
2: $\mathcal{P}_Z :=$ set of matchings P_Z covering Z such that for every edge $(p, q) \in P_Z$: $Z \cap \{p, q\} \neq \emptyset$
3: **for** $P_Z \in \mathcal{P}_Z$ **do**
4: **if** P_Z is not popular in $G' = G[Z \cup P_Z(Z) \cup U]$ **then continue**
5: **if** there is no blocking edge for P_Z in G' **then continue**
6: **if** a complete stable matching S exists on $V \setminus V'$ fulfilling the conditions in Claim 3 **then return** $P_Z \cup S$
7: **return** 'NO'

Proof. The augmenting path or cycle P_Z admits in G'_{P_Z} also exists in G_P if $P_Z \subseteq P$, since G'_{P_Z} is a subgraph of G_P. □

4.2 Second test

Next we turn to the second test in line 5. The key observation is that edges blocking P cannot occur anywhere in the graph, but only between vertices that P matches to Z. This observation is a corollary of the following claim.

Claim 2. Let P be a popular matching that leaves exactly U uncovered. If edge $(p, q) \in P$ can be reached on an alternating path in the graph G_P from an edge that blocks P such that q is the last vertex on that path, then q is in Z.

Proof. We indirectly suppose that there is an edge $(p, q) \in P$ that can be reached on an alternating path in G_P from a blocking edge such that q is the last vertex on that path, yet $q \notin Z$. If $q \notin Z$ then q must be in $N(U)$, since no vertex in U is matched in P. Now we take the alternating path in G_P that traverses (p, q). We lengthen this path past q toward U along the $(+, -)$ edge that connects q to some $u \in U$. With this new path we have shown that to P, there is an alternating path that traverses a blocking edge and ends in an unmatched vertex, which contradicts the assumption on the popularity of P. □

Corollary 4.1. *Let P be a popular matching that leaves exactly U uncovered. Each edge blocking P connects two vertices that P matches to vertices in Z.*

4.3 Third test

For better readability, we introduce the notation $V' = Z \cup P_Z(Z) \cup U$ for the vertex set of graph G'. Notice that $V \setminus V' = N(U) \setminus P_Z(Z)$, which is the set of vertices that are adjacent to at least one vertex in U and not covered by P_Z. Let us denote by D the

set of 'dangerous' vertices. These are vertices that can be reached on an alternating path from a blocking edge in G'_{P_Z} such that counting from the blocking edge, z is the further end of the matching edge $(P(z), z)$ on the path. Note that from Claim 2 we get that $D \subseteq Z$.

Dangerous vertices play a crucial role in our third test in line 6. These vertices can be reached on an alternating path from a blocking edge, which path must be discontinued in G_P before it reaches U. Our next claim guarantees that we extend P_Z to a P that fulfills this criterion, provided it is possible for the given P_Z at all.

Claim 3. Assume that P_Z is popular but not stable in G'. In G there is a popular matching P that contains P_Z and leaves exactly U uncovered if and only if $P \setminus P_Z$ is a stable matching S in $G[V \setminus V']$, covers exactly the vertices in $V \setminus V'$, and induces the following edge labeling with respect to $P = P_Z \cup S$:

1. $(+, -)$ for edges incident to U, with a $'+'$ at the vertex in U,
2. $(-, -)$ for edges between D and $V \setminus V'$,
3. $(+, -)$ for an edge (v', x) where $v' \in V' \setminus (D \cup U)$, $x \in V \setminus V'$, and v' prefers x to $P(v')$, with a $'+'$ at v'.

Proof. Let us first assume that in G there is a popular matching P that contains P_Z and leaves exactly U uncovered. Since $V \setminus V' = N(U) \setminus P_Z(Z)$ and P_Z must cover all vertices in Z by construction, $S = P \setminus P_Z$ must cover exactly the vertices in $V \setminus V'$. We will first show that S is a stable matching in $G[V \setminus V']$, and then prove that P induces an edge labeling satisfying the three points above.

The stability of S is easy to show. We have seen in Corollary 4.1 that in a popular matching every blocking edge is adjacent to two edges in P_Z, hence $S = P \setminus P_Z$ is stable in $G[V \setminus V']$.

We now check the edge labeling property in each point separately.

1. Edges incident to U trivially have a $'+'$ at the vertex in U. The other label must be a $'-'$, otherwise we found a blocking edge at an unmatched vertex, which contradicts the popularity of P.

2. Assume indirectly that there are vertices $v \in V \setminus V'$ and $z \in D$ such that (v, z) is not labelled $(-, -)$ with respect to P. By the definition of V', there exists a vertex $u \in U$ such that $\langle u - P(v) - v - z \rangle$ is an augmenting path in G_P, furthermore by the definition of D, a blocking edge is reachable from z via an alternating path in G_P. This latter path is disjoint from $(v, P(v))$, because a blocking edge can only occur in $G[Z \cup P_Z(Z)]$ (see Corollary 4.1) and from $z \in D$ it can only be reached via an alternating path that also runs in this subgraph, while both v and $P(v)$ are in $V \setminus V'$. The concatenation of these two paths is therefore an augmenting path to P in G_P, which contradicts the popularity of P.

3. Since v' prefers x to $P(v')$, a $'+'$ at v' is guaranteed. Assume indirectly that there are vertices $v' \in V' \setminus (D \cup U)$ and $x \in V \setminus V'$ such that (v', x) blocks P. Then by the definition of $V \setminus V'$, there is a vertex $u \in U$ such that $\langle u - P(x) - x - v' - P(v') \rangle$ is an augmenting path in G_P, which contradicts the popularity of P.

We now turn to proving the opposite direction. Assume that S is stable in $G[V \setminus V']$, it covers exactly $V \setminus V'$, and the three points on edge labeling are fulfilled. Since $(Z \cup P_Z(Z)) \cup (V \setminus V') = V \setminus U$, matching $P = P_Z \cup S$ leaves exactly U uncovered. We will next utilize Theorem 2.1, the characterization of popular matchings by Huang and Kavitha [19], to show that P is popular in G.

1. G_P admits no alternating cycle that contains a blocking edge.
   Neither P_Z, nor S creates an alternating cycle with a blocking edge in the subgraphs G_{P_Z} on the vertex set $Z \cup P_Z(Z)$ and G_S on the vertex set $V \setminus V'$, respectively. Therefore, for such an alternating cycle in G_P, edges between the two sets of vertices must connect paths to form a cycle. However, our three points enforce that no blocking edge leaves $V \setminus V'$. Therefore, the blocking edge must be in G_{P_Z}. These blocking edges can only be reached from dangerous vertices on an alternating path in G_P, and due to point 2 above, those paths are all disrupted by a $(-,-)$ edge between V' and $V \setminus V'$.

2. G_P admits no alternating path that contains at least two blocking edges.
   As we have argued above, all blocking edges in G_P must also block P_Z. However, an entire alternating path with two blocking edges cannot run in V', because it would contradict the popularity of P_Z. For an alternating path containing a blocking edge, leaving V' is also not an option, because of point 2 above.

3. G_P admits no alternating path that contains a blocking edge and starts at a vertex in U.
   Similarly as above, the blocking edge must also block P_Z, and the alternating path would have to leave V', which is impossible due to point 2 above.

With this we have shown that all three points in Theorem 2.1 are fulfilled by $P = P_Z \cup S$. □

4.4 Implementation of the third test

We now sketch the main idea on how to enforce the three points on edge labeling in Claim 3. For a pseudocode, please consult Algorithm 2.

The two $'+'$ signs among the 3×2 edge labels are fulfilled trivially. Three of the 4 $'-'$ signs impose a requirement on vertices in $V \setminus V'$, while the fourth one imposes a requirement on vertices in D. This latter one can be found in point 2, which requires $'-'$ votes for vertices in D on edges between D and $V \setminus V'$. Hence if a vertex $z \in D$ votes $'+'$ for a neighbor $x \in V \setminus V'$ then the stable matching S described in Claim 3 cannot exist. We test this property first, as it does not depend on how P_Z is completed to P. This constitutes the first phase of our subroutine.

For the completion of P_Z, we ensure the remaining three $'-'$ signs in three edge deletion rounds of the second phase as follows. For a vertex x in $V \setminus V'$, each point determines a list of edges incident to x such that the vote of x should be a $'-'$ for that edge with respect to P. Each of these requirements imposes an upper bound on the rank of possible partners for each vertex in $V \setminus V'$. Our key step is to delete all the

edges that are ranked worse than any of these bounds. If there is a stable matching in the remaining edge set covering every vertex in $V \setminus V'$, then this is a suitable matching S, because none of the deleted edges would have blocked it, as every deleted edge has at least one vertex where the bound, and thus the matching partner is higher in the preferences than that edge.

Algorithm 2 Subroutine checking if a complete stable matching S exists on $V \setminus V'$ fulfilling the conditions in Claim 3

Input: A popular roommates problem instance and an initial matching P_Z that has passed the first two tests.
Output: A complete stable matching S on $V \setminus V'$ or a 'NO' answer.
for $z \in D$ and $x \in N(z) \cap (V \setminus V')$ **do**
  if $x \succ_z P_Z(z)$ **then return** 'NO'
for $u \in U$ **do**
  for $x \in N(u) \cap (V \setminus V')$ **do**
    delete all (x, y) edges in $G[V \setminus V']$ such that $u \succ_x y$;
for $z \in D$ **do**
  for $x \in N(z) \cap (V \setminus V')$ **do**
    delete all (x, y) edges in $G[V \setminus V']$ such that $z \succ_x y$;
for $v \in V' \setminus (D \cup U)$ **do**
  for $x \in N(v) \cap (V \setminus V')$ such that $x \succ_v P_Z(v)$ **do**
    delete all (x, y) edges in $G[V \setminus V']$ such that $v \succ_x y$;
if a complete stable matching S exists in the remaining graph $G[V \setminus V']$ **then return** S
return 'NO'

The correctness of Algorithm 2 is proven by the following claim.

Claim 4. Assume that we are given a matching P_Z that has passed the first two tests of Algorithm 1 and the first phase of the third test. A matching S in the graph $G[V \setminus V']$ is stable, covers all vertices in $V \setminus V'$, and induces the three-point edge-labeling criteria phrased in Claim 3 if and only if it is a stable matching that covers all vertices in $V \setminus V'$ in the graph we get after deleting each edge $(x, y) \in (V \setminus V') \times (V \setminus V')$ that fulfills any of the following points.

1. $\exists u \in U$: $u \succ_x y$
2. $\exists z \in D$: $z \succ_x y$
3. $\exists v \in V' \setminus (D \cup U)$: $x \succ_v P_Z(v)$ and $v \succ_x y$

Proof. The condition on the set of covered vertices in the transformed graph is clearly necessary and sufficient, and it is straightforward that the matching in the new instance must be stable to ensure stability in the instance before edge deletions. We now revisit the three points in Claim 3 one-by-one and translate them into equivalent edge deletions. Claim 3 states that for each vertex $x \in V \setminus V'$, matching P must fulfill the following criteria:

1. (u, x) is a $(+, -)$ edge for $\forall u \in U$,
2. (z, x) is a $(-, -)$ edge for $\forall z \in D$,
3. (v, x) is a $(+, -)$ edge for $\forall v \in V' \setminus (D \cup U)$ such that $x \succ_v P(v)$.

Notice that the $'+'$ signs are unavoidable, while the first phase of Algorithm 2 guarantees the first $'-'$ sign in point 2. Thus we only need to make sure that the conditions imposed by the remaining three $'-'$ signs are fulfilled. These state that each $x \in V \setminus V'$ is matched to a partner who is ranked better than each of 1) $u \in U$, 2) $z \in D$, and 3) $v \in V' \setminus (D \cup U)$ such that $x \succ_v P_Z(v)$. These conditions enforce that all edges that are ranked worse by x than any of these three bounds need to be deleted to guarantee the three $'-'$ signs. In other words, an edge $(x, y) \in (V \setminus V') \times (V \setminus V')$ needs to be deleted if it fulfills any of the three points listed in Claim 4.

It is easy to see that if these three types of edge deletions are executed, then the edge labeling from Claim 3 is guaranteed. All that remains is to show is that stable matchings in the new instance are also stable in the original one. If such a matching M is blocked by an edge (x, y), then (x, y) must have been removed in the edge deletion round. However, due to the cover constraint on $V \setminus V'$, M must match at least one of x, y along an edge that is ranked better than (x, y). □

4.5 Correctness and running time

Theorem 4.2. *For an independent vertex set U and $Z = V \setminus N(U) \setminus U$, a popular but not stable matching that leaves exactly the vertices in U uncovered or a proof for its non-existence is outputted by Algorithm 1 in $O(n^{|Z|})(m + |Z|^3)$ time.*

Proof. We first prove that if Algorithm 1 terminates with a matching, then it is popular and leaves exactly the vertices in U uncovered. Let $P = P_Z \cup S$ be the outputted matching. By definition, P_Z covers the vertices in $Z \cup P_Z(Z)$, while S covers the vertices in $V \setminus V' = N(U) \setminus P_Z(Z)$, which is altogether $Z \cup N(U) = V \setminus U$. The popularity follows from Claim 3.

For the other direction we assume that P is a popular, but unstable matching in G, and that it leaves exactly U uncovered. Algorithm 1 tests all P_Z matchings that cover Z and are inclusion-wise minimal with respect to this property. We know from Corollary 4.1 that P is blocked by an edge that connects two vertices in $Z \cup P_Z(Z)$ for such a P_Z matching. This P_Z must therefore be generated in line 2, and pass the tests in lines 4 and 5. Furthermore, Claim 3 shows that $S = P \setminus P_Z$ exists, which then guarantees that line 6 is also passed for this particular P_Z. Note that more than one suitable stable matching might exist—our algorithm only outputs one stable matching that extends P_Z to a popular matching, which might be different from P itself.

Running time. We assume that the input is given in the form of a vertex set marking U and the strictly ordered list of edges at each vertex, in which list we assume comparison can be done in constant time. The lists also provide an edge list for the whole graph. The running time of Algorithm 1 is determined by the following factors.

1. Line 1: determine Z.
   This takes $O(m)$, because it can be done by reading the list of edges once and deleting a vertex from V if it is in U, or adjacent to any vertex in U.

2. Line 2: determine $\mathcal{P}_Z$ to Z.
   Choosing a partner for each vertex in Z can be done in $O\left(\binom{n}{|Z|}\right)$ ways. Each of the feasible P_Z matchings constructed in this step needs to be investigated separately (line 3).

3. Line 4: check the popularity of P_Z in G'.
   We need to construct the edge labeling with respect to P_Z in $O(m)$ time and then check whether P_Z is indeed a maximum weight matching in the instance. This can be done in $O(|Z|^3)$ time [36, Theorem 26.2].

4. Line 5: check for blocking edges in G'.
   The labeling in the previous step already locates all these blocking edges.

5. Line 6: checking the conditions in Claim 3 by calling Algorithm 2.

   - Determining D can be done by growing alternating trees starting from each blocking edge in G'_{P_Z}. There are at most $O(|Z|)$ end vertices of blocking edges, and even checking all paths starting from them can be done in $O(|Z|^3)$ time [36, Section 24.2a].
   - Testing in the first phase of Algorithm 2 can be done in constant time. Also, based on our three points in Claim 4, for each edge it can be decided in constant time whether it should be deleted or not.
   - Irving's algorithm [22] can find a stable matching or a proof for its non-existence in the transformed graph in $O(m)$ time. If the found stable matching does not cover all vertices in $V \setminus V'$, then no stable in this instance does, due to the Rural Hospitals Theorem [16, Theorem 4.5.2].

In total, this yields the following running time.

$$O\left(m + \binom{n}{|Z|} \cdot (m + |Z|^3 + |Z|^3 + m + m)\right) \longrightarrow O\left(n^{|Z|} \cdot (m + |Z|^3)\right) \qquad \square$$

Algorithm 1 can be utilized on a larger scale, as the following theorem shows.

Theorem 4.3. *If graph G has an odd n and $\deg(v) \geq n - c$ for all $v \in V$ and some constant c, the repeated running of Algorithm 1 can output a popular matching or a proof for its non-existence in $O\left(n^c \cdot (m + c^3)\right)$ time.*

Proof. Algorithm 1 can test on any instance of the popular roommates problem whether there is a popular matching that leaves a given non-empty independent vertex set U uncovered. In graphs on an odd number of vertices, all matchings leave a non-empty set of vertices uncovered. Therefore, if n is odd, we can iterate through all possible U sets and derive whether a popular matching exists in the instance at all. For an odd n

and $\deg(v) \geq n - c$ for all $v \in V$ and some constant c, from the definition of Z follows that $|Z| < c$. Since no vertex $u \in U$ is adjacent to a vertex in $U \cup Z$, we have that $|U \cup Z| \leq c$. Thus, the running time for checking the existence of a popular matching is polynomial for a constant c, as the following calculation demonstrates.

$$O\left(\sum_{i=1}^{c} \binom{n}{i} \cdot n^{c-i} \cdot (m + (c-i)^3)\right) \to O\left(n^c \cdot (m + c^3)\right) \qquad \square$$

We also remark that by iterating through all possible sets of uncovered vertices in the order of $|U|$, our algorithm is able to find a maximum size popular as well in graphs with an odd n and $\deg(v) \geq n - c$ for all $v \in V$. For graphs with an even n, our algorithm is only able to decide whether a non-perfect popular matching exists. This is not surprising, because deciding whether a (perfect) popular matching exists in a complete graph with an even n is NP-complete [7], while our algorithm runs in polynomial time if $|U|$ is small and the minimum degree in the graph is large.

5 Example

We demonstrate our algorithm on a chosen P_Z in the instance depicted in Figure 2. Here we only discuss the case $U = \{b\}$, $P_Z = \{(e, f), (g, h)\}$. A fully detailed example that checks the existence of a popular matching in the same instance can be found in the appendix.

a : d, b, e
b : a, d
d : b, a, e, h
e : g, a, d, f
f : g, e
g : e, h, f
h : g, d

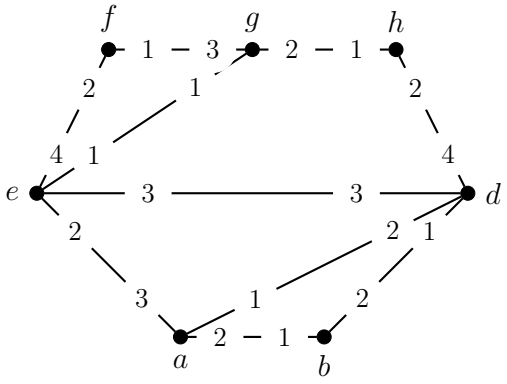

Figure 2: An example instance with no stable matching and no popular matching.

For this P_Z, the graph $G' = G[Z \cup P_Z(Z) \cup U]$ is defined by vertices $V' = \{b, e, f, g, h\}$ and edges $\{(e, f), (e, g), (f, g), (g, h)\}$. The first test in Algorithm 1 checks whether P_Z is popular in G'. As only one edge in G', namely (e, g), blocks P_Z and from this edge there is no alternating path or cycle in G'_{P_Z}, based on the characterization in Theorem 2.1 we can conclude that P_Z is popular in G'. We remark that in practice, the fulfillment of the characterization is checked such that the three points are converted into the weight requirement in a weighted matching instance defined specifically for P_Z—see Section 2. The second test in Algorithm 1 only requires the existence of a blocking edge for P_Z in G', which is granted by (e, g). The first two tests are thus passed by P_Z.

Consider now the third test in Algorithm 1. For this test we build the set of remaining vertices $V \setminus V' = \{a, d\}$ and the set of dangerous vertices $D = \{f, h\}$. Vertex f has no neighbor in $V \setminus V'$, h is adjacent to d, and $P_Z(h) \succ_h d$, so the stable matching (a, d) passes the first phase of Algorithm 2 in lines 1-2. However, in line 5 of Algorithm 2 this edge is deleted, because $d \in N(b) \cap (V \setminus V')$ and $b \succ_d a$. Therefore the remaining graph on $V \setminus V'$ is the empty graph on the two vertices a and d, and there is no complete matching that covers these two vertices after the edge deletion steps, so Algorithm 2 returns the answer 'NO'.

We conclude that this P_Z fails the third test in Algorithm 1 because on $V \setminus V'$ there is no complete stable matching after the edge deletions.

6 Computational study

We tested our algorithm on various randomly generated instances. For each combination of $n \in \{7, 9, 11, 13\}$ and $c \in \{3, 4, 5, 6\}$, 1 million graphs of minimum degree $n - c$ were generated using the Erdős–Rényi model [11]. Each edge was added with probability $p = 0.8$, and only those graphs were kept that met the bound $n - c$ on the minimum degree and indeed had at least one vertex of degree exactly $n - c$. The preference list of each vertex was then generated by choosing uniformly at random among all permutations of its edges.

We first ran Irving's algorithm [22] on each generated instance to test whether it admits a stable matching, and for those with no stable matching, we also ran our algorithm for all odd-cardinality independent vertex sets U with $|U| \leq c - 2$ to test whether there is a popular matching leaving exactly U uncovered. These tests delivered the number of instances with a stable (and thus popular) matching, and the number of instances with a popular, but no stable matching among the 1 million tested instances. The entries in Table 1 are calculated from these numbers for all tested (n, c) combinations.

Focusing on one row of the table, one can observe that for a fixed c and increasing n, the number of instances that admit a popular matching decreases. A reason for this might be that the probability of an augmenting path or cycle (see Theorem 2.1) increases as the graph grows. Fixing n and increasing c leads to more instances that admit a popular matching. Analogously to the previous reasoning, this might happen because allowing more sparse graphs gives less room for augmenting paths or cycles. In general, we can observe that for randomly generated graphs with a high minimum degree, very few instances occur that admit a popular matching but no stable matching. As the minimum degree requirement is gradually relaxed, their number, and also their ratio among instances that admit a popular matching at all, increases. We remind the reader that for an odd n and $c \leq 2$, a matching is popular if and only if it is stable [7].

The experiments were run on a standard desktop computer powered by Intel i5-3470 CPU running at 3.6GHz and 20 GiB of RAM. We used the LEMON Graph Library [8] implementation of the maximum weight perfect matching algorithm, which is based on the blossom algorithm of Edmonds [9]. Our source code is available at [37]. For the $n = 7, c = 3$ setting, deciding the existence of a popular matching took 2.6 seconds

	$n = 7$	$n = 9$	$n = 11$	$n = 13$
$c = 3$	615468 146	491189 32	401485 10	333528 3
$c = 4$	702555 1415	551617 216	446225 38	366993 9
$c = 5$	796284 8195	616446 914	493478 148	404385 26
$c = 6$	894929 32054	686405 3595	544013 478	442554 84

Table 1: The tested n and c values are organized in the different columns and rows. Each cell contains the number of instances that admit a popular matching as the top entry and the number of instances that admit a popular, but no stable matching as the bottom entry. The number of instances that admit a stable matching is the difference of the two numbers in each cell. These numbers are consistent with results measured in earlier studies [22, 34].

for those instances that admit no stable matching out of the 1 million instances in total. For comparison, we generated all matchings on the same set of instances and checked each one of them for popularity, which took 15 seconds. For $n = 11, c = 3$, these runtimes increased to 400 seconds and close to 200 hours, respectively. These numbers demonstrate that for graphs with a high minimum degree, our algorithm is indeed much faster than brute-forcing through all matchings to check whether the instance with no stable matching admits a popular matching.

7 Open questions

The most prominent future direction might be to accelerate our algorithm to reach a fixed parameter tractability result for the discussed case or to discover further polynomially solvable instance classes of the popular roommates problem. As argued in Section 4.5, graphs with a high minimum degree can only be a subject of such investigation if combined with an odd n, which is the case we covered in this paper. However, other graph parameters might prove to be fruitful. Parameterized complexity results on other popular matching problems restrict the variability of preferences [21, 27] or operate on graphs with a bounded treewidth [12].

Decomposing a popular matching to a set of edges that is stable on their subgraph and to a set of edges that is popular on their subgraph appears in two further papers. Cseh and Kavitha [7] identified the set of edges that can appear in a popular matching on a bipartite instance by showing that any popular matching M can be decomposed as $M = M_0 \cup M_1$, where M_0 is a so-called *dominant* matching in the subgraph induced by the vertices matched in M_0, and in the subgraph induced by the remaining vertices, M_1 is stable. Kavitha [25] searched for popular matchings in non-bipartite instances by showing that every popular matching can be partitioned into a stable part and a *truly popular* part. Discovering a connection between these three decompositions

might lead to new structural insights.

References

[1] D. J. Abraham, R. W. Irving, T. Kavitha, and K. Mehlhorn. Popular matchings. *SIAM Journal on Computing*, 37:1030–1045, 2007.

[2] P. Biró, R. W. Irving, and D. F. Manlove. Popular matchings in the marriage and roommates problems. In *Proceedings of CIAC '10: the 7th International Conference on Algorithms and Complexity*, volume 6078 of *Lecture Notes in Computer Science*, pages 97–108. Springer, 2010.

[3] K. S. Chung. On the existence of stable roommate matchings. *Games and Economic Behavior*, 33(2):206–230, 2000.

[4] M. Condorcet. *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. L'Imprimerie Royale, 1785.

[5] Á. Cseh. Popular matchings. *Trends in Computational Social Choice*, 105, 2017.

[6] Á. Cseh and T. Kavitha. Popular matchings in complete graphs. In *38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, 2018.

[7] Á. Cseh and T. Kavitha. Popular matchings in complete graphs. *Algorithmica*, 83(5):1493–1523, 2021. Earlier version appeared as [6].

[8] B. Dezső, A. Jüttner, and P. Kovács. LEMON – an open source C++ graph template library. *Electronic Notes in Theoretical Computer Science*, 264(5):23–45, 2011. Proceedings of the Second Workshop on Generative Technologies (WGT) 2010.

[9] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[10] E. Erdem, M. Fidan, D. Manlove, and P. Prosser. A general framework for stable roommates problems using answer set programming. *Theory and Practice of Logic Programming*, 20(6):911–925, 2020.

[11] P. Erdős, A. Rényi, et al. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[12] Y. Faenza, T. Kavitha, V. Powers, and X. Zhang. Popular matchings and limits to tractability. In *SODA'19*, pages 2790–2809, 2019.

[13] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69:9–15, 1962.

[14] P. Gärdenfors. Match making: assignments based on bilateral preferences. *Behavioural Science*, 20:166–173, 1975.

[15] S. Gupta, P. Misra, S. Saurabh, and M. Zehavi. Popular matching in roommates setting is NP-hard. *ACM Transactions on Computation Theory (TOCT)*, 13(2):1–20, 2021.

[16] D. Gusfield and R. W. Irving. *The Stable Marriage Problem: Structure and Algorithms*. MIT Press, 1989.

[17] C.-C. Huang and T. Kavitha. Popular matchings in the stable marriage problem. In *Proceedings of ICALP '11: the 38th International Colloquium on Automata, Languages and Programming*, volume 6755 of *Lecture Notes in Computer Science*, pages 666–677. Springer, 2011.

[18] C.-C. Huang and T. Kavitha. Near-popular matchings in the roommates problem. *SIAM Journal on Discrete Mathematics*, 27(1):43–62, 2013.

[19] C.-C. Huang and T. Kavitha. Popular matchings in the stable marriage problem. *Information and Computation*, 222:180–194, 2013.

[20] C.-C. Huang and T. Kavitha. Popularity, mixed matchings, and self-duality. *Mathematics of Operations Research*, 46(2):405–427, 2021.

[21] C.-C. Huang, T. Kavitha, D. Michail, and M. Nasre. Bounded unpopularity matchings. *Algorithmica*, 61(3):738–757, 2011.

[22] R. W. Irving. An efficient algorithm for the "stable roommates" problem. *Journal of Algorithms*, 6:577–595, 1985.

[23] T. Itoh and O. Watanabe. Weighted random popular matchings. *Random Structures and Algorithms*, 37(4):477–494, 2010.

[24] T. Kavitha. Max-size popular matchings and extensions. *arXiv preprint arXiv:1802.07440*, 2018.

[25] T. Kavitha. Popular roommates in simply exponential time. In *39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[26] T. Kavitha, J. Mestre, and M. Nasre. Popular mixed matchings. *Theoretical Computer Science*, 412:2679–2690, 2011.

[27] A. M. Krishnapriya, M. Nasre, P. Nimbhorkar, and A. Rawat. How good are popular matchings? In *17th International Symposium on Experimental Algorithms (SEA 2018)*, volume 103, page 9. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018.

[28] M. Mahdian. Random popular matchings. In J. Feigenbaum, J. C. Chuang, and D. M. Pennock, editors, *Proceedings of EC '06: the 7th ACM Conference on Electronic Commerce*, pages 238–242. ACM, 2006.

[29] D. F. Manlove. *Algorithmics of Matching Under Preferences*. World Scientific, 2013.

[30] S. Mertens. Stable roommates problem with random preferences. *Journal of Statistical Mechanics: Theory and Experiment*, 2015(1):P01020, 2015.

[31] C. Palmer and D. Pálvölgyi. At most 3.55^n stable matchings. *arXiv preprint arXiv:2011.00915*, 2020.

[32] B. Pittel. The "Stable Roommates" problem with random preferences. *Annals of Probability*, 21(3):1441–1477, 1993.

[33] B. G. Pittel and R. W. Irving. An upper bound for the solvability probability of a random stable roommates instance. *Random Structures and Algorithms*, 5:465–486, 1994.

[34] P. Prosser. Stable roommates and constraint programming. In *International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems*, pages 15–28. Springer, 2014.

[35] S. Ruangwises and T. Itoh. Random popular matchings with incomplete preference lists. In *International Workshop on Algorithms and Computation*, pages 106–118. Springer, 2018.

[36] A. Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003.

[37] Tools for measuring stable and popular matchings sizes. `https://github.com/atimaly/Stable-and-Popular-Matchings`, 2021. Accessed 15 July 2021.

A Examples

We now list two examples, one NO-instance and one YES-instance.

A.1 NO-instance

We demonstrate a full run of our algorithm on the instance depicted in Figure 2. It is easy to see that the instance admits no stable matching, because each stable matching would have to include the edge (e, g), and after fixing this edge, we are left with a preference cycle of length 3 on vertices $\{a, b, d\}$, which proves the non-existence of a stable matching.

Since n is odd, each popular matching must leave an odd number of vertices uncovered. In order to check whether there is a popular matching, we now iterate

Case	P_Z	$V(G')$	$E(G')$	test
$U = \{a\}$	$(f,g),(d,h)$	a,d,f,g,h	$(a,d),(d,h),(f,g),(g,h)$	1 (A)
$Z = \{f,g,h\}$	$(e,f),(g,h)$	a,e,f,g,h	$(a,e),(e,f),(e,g),(f,g),(g,h)$	1 (B)
$U = \{b\}$	$(e,f),(g,h)$	b,e,f,g,h	$(e,f),(e,g),(f,g),(g,h)$	3 (C)
$Z = \{e,f,g,h\}$	$(a,e),(f,g),(d,h)$	$V(G)$	$E(G)$	1 (D)
$U = \{d\}$	(f,g)	d,f,g	(f,g)	2 (E)
$Z = \{f,g\}$	$(e,f,(g,h)$	d,e,f,g,h	$(d,e),(d,h),(e,f),(e,g),(f,g),(g,h)$	1 (F)
$U = \{e\}$	$(a,b),(d,h)$	a,b,d,e,h	$(a,b),(a,d),(a,e),(b,d),(d,e),(d,h)$	1 (F)
$Z = \{b,h\}$	$(a,b),(g,h)$	a,b,e,g,h	$(a,b),(a,e),(e,g),(g,h)$	1 (G)
	$(b,d),(g,h)$	a,b,e,g,h	$(a,b),(a,e),(e,g),(g,h)$	1 (G)
$U = \{f\}$	$(a,b),(d,h)$	a,b,d,f,h	$(a,b),(a,d),(b,d),(d,h)$	3 (H)
$Z = \{a,b,d,h\}$	$(a,b),(d,e),(g,h)$	$V(G)$	$E(G)$	1 (I)
	$(a,e),(b,d),(g,h)$	$V(G)$	$E(G)$	1 (J)
$U = \{g\}$	$(a,b),(d,e)$	a,b,d,e,g	$(a,b),(a,d),(a,e),(b,d),(d,e),(e,g)$	1 (G)
$Z = \{a,b,d\}$	$(a,b),(d,h)$	a,b,d,g,h	$(a,b),(a,d),(b,d),(d,h),(g,h)$	1 (G)
	$(a,e),(b,d)$	a,b,d,e,g	$(a,b),(a,d),(a,e),(b,d),(d,e),(e,g)$	1 (G)
$U = \{h\}$	$(a,b),(e,f)$	a,b,e,f,h	$(a,b),(a,e),(e,f)$	2 (E)
$Z = \{a,b,e,f\}$	$(a,b),(d,e),(f,g)$	$V(G)$	$E(G)$	1 (K)
	$(a,e),(b,d),(f,g)$	$V(G)$	$E(G)$	1 (L)
$U = \{a,f,h\}$	$\varnothing$	a,f,h	$\varnothing$	3 (M)
$Z = \varnothing$				
$U = \{b,e,h\}$	$\varnothing$	b,e,h	$\varnothing$	3 (N)
$Z = \varnothing$				
$U = \{b,f,h\}$	$\varnothing$	b,f,h	$\varnothing$	3 (O)
$Z = \varnothing$				

Table 2: The first column contains the tested U set and the set of vertices Z, calculated from U. In the second column, we list the possible P_Z matchings. To each of these, the vertices and edges of G' are listed. Finally, we mark which test the current P_Z failed and give a detailed explanation of this failure in a separate list.

through all vertex sets U with an odd cardinality and no spanned edge, and apply Algorithms 1 and 2 to determine whether there is a popular matching that leaves exactly the vertices in U uncovered. Table 2 summarizes these steps. As the instance admits no popular matching, each tested P_Z fails at some point. The exact explanation on how this happens is deferred to the list marked with capital letters.

A: P_Z is not popular in G', because the blocking edge (g,h) can be reached from a in G'_{P_Z}.

B: P_Z is not popular in G', as (a,e) blocks P_Z.

C: Edge (e,g) blocks P_Z and $D = \{f,h\}$. The graph spanned by $V \setminus V' = \{a,d\}$ consists of the edge (a,d) itself, but even this edge is deleted in Algorithm 2. We conclude that after the edge deletions in the third test, no stable matching covers the entire $V \setminus V'$.

D: P_Z is not popular in G', as (b, d) blocks P_Z.

E: P_Z is stable in G'.

F: P_Z is not popular in G', as (d, e) blocks P_Z.

G: P_Z is not popular in G', as (e, g) blocks P_Z.

H: Edge (a, d) blocks P_Z and $D = \{b, h\}$. The graph spanned by $V \setminus V' = \{e, g\}$ consists of the edge (a, d) itself, but even this edge is deleted in Algorithm 2, because $g \succ_h d$. We conclude that after the edge deletions in the third test, no stable matching covers the entire $V \setminus V'$.

I: P_Z is not popular in G', because the blocking edge (a, d) can be reached from f in G'_{P_Z}.

J: P_Z is not popular in G', because the blocking edge (a, b) can be reached from f in G'_{P_Z}.

K: P_Z is not popular in G', because the blocking edge (e, g) can be reached from h in G'_{P_Z}.

L: P_Z is not popular in G', because the blocking edge (a, b) can be reached from h in G'_{P_Z}.

M: In Algorithm 2 we delete (b, d) since $a \succ_b d$. After this deletion, no matching covers $V \setminus V' = \{b, d, e, g\}$.

N: In Algorithm 2 we delete (a, d) since $b \succ_d a$. After this deletion, no matching covers $V \setminus V' = \{a, d, f, g\}$.

O: In Algorithm 2 we delete (a, d) since $b \succ_d a$. After this deletion, no matching covers $V \setminus V' = \{a, d, e, g\}$.

A.2 YES-instance

The instance depicted in Figure 3 admits a popular matching, but no stable matching. Since n is odd, each popular matching must leave an odd number of vertices uncovered. In order to check whether there is a popular matching, we start iterating through all vertex sets U with an odd cardinality and no spanned edge, and apply Algorithms 1 and 2 to determine whether there is a popular matching that leaves exactly the vertices in U uncovered. Table 3 summarizes these steps. A popular matching consisting of edges $(a, b), (d, h), (e, g)$ is found in the line marked by (*). The algorithm then terminates with outputting this matching.

$a:$ $e,$ b
$b:$ $e,$ $h,$ $g,$ $d,$ $a,$ f
$d:$ $h,$ $e,$ b
$e:$ $d,$ $h,$ $b,$ $g,$ $a,$ f
$f:$ $b,$ e
$g:$ $e,$ $b,$ h
$h:$ $g,$ $e,$ $d,$ b

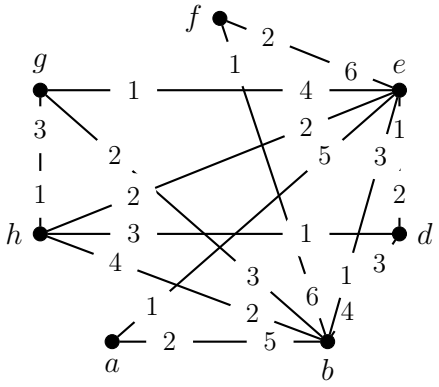

Figure 3: Example instance on 7 vertices that admits a popular matching but no stable matching.

Case	P_Z	$V(G')$	$E(G')$	test	reason for test failure (if any)
$U = \{a\}$	$(b,d),(e,f),(g,h)$	$V(G)$	$E(G)$	1	(a,e) blocks P_Z
$Z = \{d,f,g,h\}$	$(b,f),(d,e),(g,h)$	$V(G)$	$E(G)$	1	(a,e) blocks P_Z
	$(b,f),(d,h),(e,g)$	$V(G)$	$E(G)$	1	(a,b) blocks P_Z
	$(b,g),(d,h),(e,f)$	$V(G)$	$E(G)$	1	(a,b) blocks P_Z
$U = \{b\}$	$\varnothing$	b	$\varnothing$	3	no complete matching in $G \setminus G'$
$Z = \varnothing$					
$U = \{d\}$	$(a,b),(e,f),(g,h)$	$V(G)$	$E(G)$	1	(d,e) blocks P_Z
$Z = \{a,f,g\}$	$(a,e),(b,f),(g,h)$	$V(G)$	$E(G)$	1	(d,e) blocks P_Z
$U = \{e\}$	$\varnothing$	e	$\varnothing$	3	no complete matching in $G \setminus G'$
$Z = \varnothing$					
$U = \{f\}$	$(a,b),(d,e),(g,h)$	$V(G)$	$E(G)$	1	the blocking edge (b,g) is reachable
$Z = \{a,d,g,h\}$					from f in G_{P_Z} via an alternating path
	$(a,e),(b,d),(g,h)$	$V(G)$	$E(G)$	1	the blocking edge (d,e) is reachable
					from f in G_{P_Z} via an alternating path
	$(a,e),(b,g),(d,h)$	$V(G)$	$E(G)$	1	the blocking edge (e,g) is reachable
					from f in G_{P_Z} via an alternating path
	$(a,b),(d,h),(e,g)$	$V(G)$	$E(G)$	-	*

Table 3: The first column contains the tested U set and the set of vertices Z, calculated from U. In the second column, we list the possible P_Z matchings. To each of these, the vertices and edges of G' are listed. Then we mark which test the current P_Z failed and explain this failure in the last column.