

Preference-constrained oriented matching*

Lisa Fleischer[†]

Zoya Svitkina[‡]

Abstract

We introduce and study a combinatorial problem called *preference-constrained oriented matching*. This problem is defined on a directed graph in which each node has preferences over its out-neighbors, and the goal is to find a maximum-size matching on this graph that satisfies a certain preference constraint. One of our main results is a structural theorem showing that if the given graph is complete, then for any preference ordering there always exists a feasible matching that covers a constant fraction of the nodes. This result allows us to correct an error in a proof by Azar, Jain, and Mirrokni [1], establishing a lower bound on the price of anarchy in coordination mechanisms for scheduling. We also show that the preference-constrained oriented matching problem is APX-hard and give a constant-factor approximation algorithm for it.

1 Introduction

We introduce a combinatorial problem called *preference-constrained oriented matching*, defined below. Given a directed graph $G = (V, E)$, a *matching* $F \subseteq E$ is a subset of edges of G such that each node is incident to at most one edge in this subset. Since the graph is directed, a matching can be expressed as a bijection $\mu : A \rightarrow B$ between two disjoint subsets of nodes $A, B \subset V$, such that $F = \{(a, \mu(a)) \mid a \in A\}$. We write $\mu(a) = b$ and $\mu^{-1}(b) = a$ whenever an edge (a, b) is in the matching. The matching is said to be *oriented* because in our setting, including an edge (a, b) in the matching is not equivalent to including the edge (b, a) .

The input to the preference-constrained oriented matching problem consists of a directed graph $G = (V, E)$ with $|V| = n$ nodes, each of which has a strict preference ordering over its out-neighbors (or, equivalently, over its outgoing edges). In particular, we write $u \succ_v w$ if both edges (v, u) and (v, w) are in E , and node v prefers u to w . The goal of the problem is to find a maximum-size oriented matching $\mu : A \rightarrow B$ subject to the following *preference constraint*: every

node $a \in A$ prefers its assigned match, $\mu(a) \in B$, to any other node $a' \in A$ to which it has an edge. More formally, $\mu(a) \succ_a a'$ for all $a \in A$ and $a' \in A$ such that $(a, a') \in E$. We say that an oriented matching is *valid* if it satisfies the preference constraint.

One of our main results is a structural theorem showing that if G is a complete directed graph, then for any preference ordering there always exists a valid oriented matching of size at least $\frac{n}{3}$ (i.e., with $|A| = |B| \geq \frac{n}{3}$). This result allows us to correct an error in a proof that appeared in [1], establishing a lower bound on the price of anarchy in coordination mechanisms for scheduling. We explain the details in Section 4. We also give an example of preferences in a complete graph for which the size of the maximum valid matching is exactly $\frac{n}{3}$, thus showing that our existence result is tight.

In addition to the structural result, we show that on a complete directed graph, a valid matching of size at least $\frac{n}{3}$ can be found in polynomial time. This immediately gives us a $\frac{2}{3}$ -approximation for the problem of finding a maximum valid oriented matching on complete graphs, since any matching can have size at most $\frac{n}{2}$. For the case that G is not necessarily a complete graph, we show that the same algorithm is a $\frac{1}{3}$ -approximation. We then further investigate the complexity of the problem, and show that it is NP-hard, and, in fact, APX-hard to approximate, even on complete graphs. We then briefly discuss a version of the problem with specified sides of the matching. In particular, the given set of nodes V consists of two disjoint subsets $V = S \cup T$, and we have an additional requirement that the produced preference-constrained oriented matching has to consist of edges in $E \cap (S \times T)$, or, in other words, to satisfy $A \subseteq S$ and $B \subseteq T$. Surprisingly, this version of the problem is substantially harder to approximate, and we give a reduction showing that it is as hard to approximate as INDEPENDENT SET.

1.1 Related work Finding maximum-cardinality matchings in graphs is a well-researched topic. Polynomial-time algorithms for finding maximum matchings in bipartite [3] as well as general [2] graphs are known. Our problem differs from these by the presence of preference constraints, but our algorithm uses the idea of augmenting paths that was first introduced

*This work was supported in part by NSF grant CCF-0728869.

[†]Department of Computer Science, Dartmouth, USA.

[‡]Department of Computing Science, University of Alberta, Canada. Supported by Alberta Ingenuity.

in the context of network flow and maximum matching.

A number of problem formulations that involve matching entities with preferences have been considered in the literature. The most well-studied of them is the stable matching problem [4, 6, 9], where the graph is bipartite, both sides have preferences, and a matching is considered stable if no two elements prefer each other to their assigned matches. For complete bipartite graphs, stable matchings always exist and can be found efficiently [4]. For the case of incomplete preferences, a stable matching of maximum size can also be found in polynomial time [5]. A version of the problem in which the graph is not bipartite is called the stable roommates problem. In this setting a stable matching does not always exist, but if it does, then one can be found in polynomial time [8]. This is also true for the case of incomplete preferences [6].

Our problem resembles these ones, but the criteria for feasibility of a matching are different. First, in our case, only the preferences of elements on one side of a proposed matching (ones in the set A) matter, whereas for stable matching or roommates, the condition violating stability involves the preferences of both sides. The second difference is that our problem is concerned with preferences of a node over other nodes on the same side as itself, whereas the stable matching is concerned with preferences over the other side, and the stable roommates problem does not distinguish between sides. Another difference is that in our model, unmatched elements cannot cause a violation of feasibility, whereas in the case of stable matching they can.

2 Existence result and the approximation algorithm

In this section we prove the following several results.

THEOREM 2.1. *If G is a complete directed graph on $n \geq 2$ nodes, then for any preference orderings of the nodes over their out-neighbors, there exists an oriented matching over G , satisfying the preference constraint, of size at least $\frac{n}{3}$. Moreover, this matching can be found in polynomial time.*

The next example shows that the bound of $\frac{n}{3}$ in Theorem 2.1 is tight.

EXAMPLE 2.1. We construct a class of graphs with preferences which do not contain valid matchings of size greater than $\frac{n}{3}$. The key observation is that if three nodes of V are each other's first choices in a cycle (u prefers v most, v prefers w , and w prefers u), then at most one of them can be in the set A in a valid matching. This is because if two of them are in A , then one of them will prefer the other to its match in the set B . The tight

example is then obtained for arbitrary n divisible by 3 by arranging the first choices to form disjoint 3-cycles.

As any matching on a graph with n nodes has size at most $\frac{n}{2}$, Theorem 2.1 immediately implies

COROLLARY 2.1. *There is a polynomial-time $\frac{2}{3}$ -approximation algorithm for the problem of finding a maximum-size preference-constrained oriented matching on a complete directed graph.*

In addition, we show that the same algorithm is also a $\frac{1}{3}$ -approximation for the case that G is not a complete graph:

THEOREM 2.2. *There is a polynomial-time $\frac{1}{3}$ -approximation algorithm for the problem of finding a maximum-size preference-constrained oriented matching on an arbitrary directed graph.*

We now present the algorithm that is used to prove Theorems 2.1 and 2.2. It iteratively builds a valid matching by starting with an empty matching μ and repeatedly calling the subroutine *Improve*(μ) (see Algorithm 1), which either increases the size of μ by one, or outputs *done*, at which point the algorithm stops. We next describe the procedure *Improve*(μ).

Procedure *Improve*(μ) keeps the n nodes of G partitioned into five disjoint sets, $A, B^\circ, B^\bullet, C, R$. The sets A and $B = B^\circ \cup B^\bullet$ are subsets of V on which the current matching μ is defined, with $\mu : A \rightarrow B$. The unmatched nodes are initially all in the set C , with $R = \emptyset$, but during the run of the algorithm they are partitioned in some way between the sets C and R . The set B is partitioned into the set B^\bullet of *marked* nodes and the set B° of *unmarked* nodes. We use the notation $X \xrightarrow{x} Y$ to represent the action of transferring a node x from set X to set Y . Directions of possible movements of elements between sets are shown in Figure 1. As a pre-processing step, we make sure that no node $a \in A$ prefers any node $c \in C$ to its match $\mu(a) = b \in B$. If there are such nodes, then swap b and c , setting $\mu(a) = c$, and putting b into set C . Note that the new matching is also valid, and the process terminates in polynomial time, since for each $a \in A$, its match only improves.

To define the algorithm, we need to introduce the notion of a *preference cycle* in the matching. A preference cycle is an even cycle that alternates between nodes in the set A and nodes in the set B of the matching, in the following way. From a node $a \in A$, it goes to its match $\mu(a) \in B$; from a node $b \in B$, it goes to a node $a' \in A$ such that a' prefers b to its current match, $b \succ_{a'} \mu(a')$. Given a preference cycle, it can be

Algorithm 1 $\text{Improve}(\mu)$. Input: valid matching $\mu : A \rightarrow B$.

```

1: Let  $B^\circ = B$ ,  $B^\bullet = \emptyset$ ,  $C = V \setminus (A \cup B)$ ,  $R = \emptyset$ .
2: Ensure that all nodes  $a \in A$  prefer  $\mu(a)$  to any  $c \in C$ 
3: loop
4:   Set all edges of  $G$  as unselected
5:   while there is  $c \in C$  with unselected outgoing edges and no selected edge from  $c$  into  $A$  do
6:     select the most-preferred unselected edge from  $c$ , say  $(c, x)$ 
7:     if  $x \in C \cup R$  then return  $\mu + (c, x)$ 
8:     else if  $x \in B^\circ$  then set  $\rho(c) = x$ ;  $B^\circ \xrightarrow{x} B^\bullet$ ;  $C \xrightarrow{c} R$ 
9:   end while
10:  if some  $c \in C$  has a selected edge to  $a \in A$  such that  $\mu(a) = b \in B^\bullet$  then
11:    return  $\mu - (a, b) + (c, a) + (\rho^{-1}(b), b)$ 
12:  else if two nodes  $c, c' \in C$  have selected edges to some  $a \in A$  then
13:    repeat
14:      find a preference cycle or an alternating path starting from  $a$ 
15:      if a preference cycle is found then eliminate it
16:      until an alternating path  $P = \{a, \dots, b\}$  is found
17:      shift the matching along the alternating path  $P$ , removing  $a$  and  $b$  from  $\mu$ 
18:      if  $b \in B^\bullet$  then return  $\mu + (c, a) + (\rho^{-1}(b), b)$ 
19:      else add  $(c, a)$  to  $\mu$ ; set  $\rho(c') = a$ ;  $C \xrightarrow{c} A$ ;  $A \xrightarrow{a} B^\bullet$ ;  $C \xrightarrow{c'} R$ ;  $B^\circ \xrightarrow{b} C$ 
20:    else return done
21:  end if
22: end loop

```

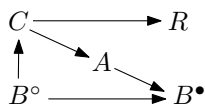


Figure 1: Transfers between sets

eliminated by a rotation procedure that matches each cycle node in A to its predecessor in B (instead of the successor to which it was previously matched). Note that this rotation improves the matching for all nodes of A that are in the cycle and preserves the validity of the matching. We also define an *alternating path* $P = \{a_1, b_1, a_2, b_2, \dots, a_k, b_k\}$ to be a sequence of nodes that starts with $a_1 \in A$, alternates between nodes of A and B in the same way as a preference cycle does, and ends with a node $b_k \in B$, with the property that the sequence cannot be extended farther from b_k . The matching can be *shifted* along an alternating path P by removing the endpoints a_1 and b_k from the matching, and assigning the remaining nodes of $P \cap A$ to be matched to their predecessors in the path. This decreases the size of the matching by one. We note that starting from any $a \in A$, either a preference cycle (that does not necessarily include a) or an alternating path (that starts at a) can be found in μ in polynomial time.

$\text{Improve}(\mu)$ increases the size of the matching in one of several ways. The simplest case is when two

nodes not participating in the current matching are found that can be matched to each other and added to μ (this happens on line 7). Alternatively, a matched pair (a, b) can be removed from the matching and replaced by two pairs (c, a) and (r, b) , with $c \in C$ and $r \in R$ (line 11). The replacement may also be more complex, when the matching is shifted along an alternating path $\{a, \dots, b\}$, and then two new pairs (c, a) and (r, b) are added (lines 17-18). However, in order to perform any of these updates, the algorithm needs to find new edges of G , such as (c, a) and (r, b) , that can be added to the matching without violating the preference constraint. For this purpose, it keeps track of *selected* outgoing edges from set C and of a mapping ρ . R is the set of *reserved* nodes that correspond to marked nodes in B , and $\rho : R \rightarrow B^\bullet$ is a bijection that captures this correspondence. The requirements satisfied by selected edges, ρ , and the matching can be stated formally as the following invariants, which, as we show in Lemma 2.2, are maintained throughout the execution of $\text{Improve}(\mu)$.

- I If a node $c \in C$ has a selected edge to $a \in A$, then c prefers a to nodes in $A \setminus \{a\}$, C , R , or B° .
- II Any node $r \in R$ prefers $\rho(r)$ to any node in A , C , or B° .
- III Any node $a \in A$ prefers its match $\mu(a)$ to any node in A , C or R .

We now show that $\text{Improve}(\mu)$ runs in polynomial time. One way to see this is that each iteration of the outer-most loop either increases the size of the matching and exits, or it adds at least one more reserved node to the set R .

LEMMA 2.1. *Procedure $\text{Improve}(\mu)$ terminates in polynomial time.*

Proof. As already mentioned, the pre-processing on line 2 can be done in polynomial time. The while loop on line 5 keeps selecting previously unselected outgoing edges from nodes in C , and may also remove nodes from C . So it ends in a polynomial number of steps. The repeat loop can find at most a polynomial number of preference cycles, after which it ends by finding an alternating path. This is because every time it eliminates a cycle, it improves the matching for the nodes in set A involved in this cycle. All other steps of the algorithm either exit or transfer some elements between sets. As the graph of possible transfers in Figure 1 is acyclic, the procedure as a whole terminates in polynomial time. \square

LEMMA 2.2. *At any point in the main loop of Algorithm 1, invariants I-III hold.*

Proof. When the loop is first entered, Invariants I and II hold vacuously. Invariant III holds with respect to set A by the assumption that the initial matching is valid, with respect to set C by the pre-processing step, and with respect to set R because R is empty.

We now show that the invariants are maintained as the algorithm proceeds. First notice that the sets $A \cup C \cup R \cup B^\circ$ and $A \cup C \cup B^\circ$ in Figure 1 don't have any incoming edges, meaning that no new elements are added to them during the course of the algorithm. This means that once Invariant I holds for a particular edge (c, a) , it will not become violated later by some node x with $x \succ_c a$ entering the set $A \cup C \cup R \cup B^\circ$. The same holds for Invariant II. Thus it suffices to establish that these invariants hold when a particular edge (c, a) is first selected, or, respectively, when $\rho(r)$ is first defined. The similar property does not hold for Invariant III, so more care is needed.

To establish Invariant I, we examine the while loop, where new edges are selected. When a node c is chosen by the while loop, it has no previously-selected edges to A (by the way it is chosen), no previously-selected edges to C or R (since the algorithm exits on line 7 whenever such an edge is selected), and no previously-selected edges to B° (since the algorithm moves c into R on line 8 whenever such an edge is selected). As the newly-selected edge (c, x) is the most-preferred one, we have that if $x \in A$, then Invariant I holds.

For Invariant II, we note that new nodes are added to R and $\rho(\cdot)$ is defined at two points in the algorithm: on lines 8 and 19. On line 8, the edge from c to x is the last one just selected, which means that c prefers $x = \rho(c)$ to any node in A (otherwise the loop would have stopped selecting edges from c), in C (otherwise the procedure would have exited), or any other node in B° . On line 19, c' is a node in C which has a selected edge to a , a node in A . So by Invariant I, c' prefers a to other nodes in A , C , or B° . Thus, when we set $\rho(c') = a$, Invariant II holds.

Invariant III can potentially become violated if either the matching μ changes, or a new node enters the set $A \cup C \cup R$. But note that any change in the matching for a node $a \in A$ that happens as a result of preference cycle elimination or the shifting of an alternating path can match a only to a more-preferred node, so these changes cannot violate our condition. Another change to the matching is the addition of the edge (c, a) to μ on line 19. But this is a selected edge, so Invariant I guarantees that c prefers a to other nodes in A , C , or R . What remains to check is that Invariant III still holds when nodes are moved between sets. The only time that a new node is added to $A \cup C \cup R$ is on line 19, when we move b from B° to C . But this does not violate the invariant because the fact that path P cannot be extended beyond b implies that no node in A prefers b to its match. \square

LEMMA 2.3. *If $\text{Improve}(\mu)$ succeeds, it outputs a valid matching of size $|\mu| + 1$.*

Proof. The resulting matching has size $|\mu| + 1$ because it is obtained either by adding a pair of nodes to the existing matching (line 7) or by removing two nodes from the matching, and then adding two pairs to it (lines 11 and 18). What remains is to show that this is valid matching.

During the execution of the algorithm, the fact that μ is a valid matching follows from Invariant III. We show that the preference constraint still holds when new matched pairs are added to the returned matching (lines 7, 11, 18). The edge (c, x) is added on line 7, where c prefers x to anything in A ; (c, a) is added on lines 11 and 18, where $c \in C$ has a selected edge to a and therefore prefers it to any node in A by Invariant I; and $(\rho^{-1}(b), b)$ is added on lines 11 and 18, where $\rho^{-1}(b) \in R$ prefers b over nodes in A by Invariant II. The nodes that are already in A prefer their matches to these newly added nodes from sets C or R by Invariant III. \square

We now prove Theorems 2.1 and 2.2.

Proof of Theorem 2.1. We show that, given a complete graph G on $n \geq 2$ nodes and a valid matching μ ,

procedure $Improve(\mu)$ returns *done* only if $|\mu| \geq \frac{n}{3}$. Together with Lemmas 2.1 and 2.3, this proves the theorem. If the input matching μ is empty, then the algorithm adds the first edge that it selects to the matching, and thus succeeds. So we consider the case that μ is non-empty.

Assume that the procedure is not able to increase the size of the matching and exits on line 20, and consider the last iteration of the outer-most loop. In particular, neither the if condition on line 10 nor the else condition on line 12 is satisfied in this iteration. Since G is a complete directed graph, by the end of the while loop, each node in the set C has exactly one selected edge to the set A . This means that there are exactly $|C|$ selected edges from C to A . Let $A^\bullet = \{a \in A \mid \mu(a) \in B^\bullet\}$ be the nodes of A matched to B^\bullet , and $A^\circ = A \setminus A^\bullet$ be the ones matched to B° . Now, there are no selected edges from C to A^\bullet , as otherwise the if condition of line 10 would be satisfied. Also, there is at most one selected edge from C to any $a \in A^\circ$, as otherwise the condition on line 12 is met. So the number of selected edges from C to A , and thus the size of C , is at most $|A^\circ| = |B^\circ|$. We also note that $|R| = |B^\bullet|$, which gives us $n = |A| + |B| + |C| + |R| \leq |A| + |B| + |B^\circ| + |B^\bullet| = 3|\mu|$, concluding the proof. \square

Proof of Theorem 2.2. Similarly to the proof above, we consider the end of the while loop in the last iteration of $Improve(\mu)$'s outer loop. Let $\mu^* : A^* \rightarrow B^*$ denote some optimal valid oriented matching. We first examine the intersection of sets A^* and C . Let $C_1 = \{c \in C \cap A^* \mid \exists a \in A \text{ s.t. } (c, a) \text{ is selected}\}$, and $C_2 = (C \cap A^*) \setminus C_1$. As in the proof of Theorem 2.1, C_1 has no selected edges to A^\bullet , and has at most one selected edge to each node in A° , as otherwise the conditions on lines 10 or 12 would be met. Thus, $|C_1| \leq |A^\circ| = |B^\circ|$. For the nodes in C_2 , all their outgoing edges are selected (since $c \in C_2$ has no edges to A , the while loop does not stop until it selects all edges from c). These selected edges of C_2 include the edges $(c, \mu^*(c))$ used by the optimal solution, but none of them go from C_2 to other nodes in C (otherwise the procedure would exit on line 7 after selecting such an edge). So if we consider the endpoints $\{\mu^*(c) \mid c \in C_2\}$, which are in B^* but not in C , we can conclude that $|(A \cup B \cup R) \cap B^*| \geq |C_2|$. But since A^* and B^* are disjoint, we have that $|(A \cup B \cup R) \cap A^*| \leq |A| + |B| + |R| - |C_2|$. Now we bound the size of A^* as follows:

$$\begin{aligned} |A^*| &= |C_1| + |C_2| + |(A \cup B \cup R) \cap A^*| \\ &\leq |B^\circ| + |C_2| + |A| + |B| + |B^\bullet| - |C_2| = 3|\mu|. \end{aligned}$$

This means that the matching μ found by the algorithm is a $\frac{1}{3}$ -approximation to the optimum. \square

Finally, we give a simple example to show that the analysis of our algorithm is tight.

EXAMPLE 2.2. Let G consist of nodes a, b, r, c_1, c_2, c_3 . The directed edges are (a, b) , (a, c_1) , (r, b) , (r, c_3) , and (c_2, b) , with preferences $b \succ_a c_1$ and $b \succ_r c_3$. In this instance, there is a valid matching of size 3, with edges (a, c_1) , (c_2, b) , and (r, c_3) . However, the algorithm may end after finding a matching of size one, namely $\mu(a) = b$, with $R = \{r\}$, $\rho(r) = b$, and $C = \{c_1, c_2, c_3\}$.

3 Hardness results

We show that the problem of finding a valid matching of maximum size is NP-hard. The reduction is from 3SAT, and the instance of the problem that we construct admits a valid matching of size $n/2$ if and only if the given SAT instance is satisfiable.

Let the given 3SAT instance contain k variables and m clauses. For each variable x_j , let t_j be the number of occurrences of the literal x_j in the formula, and f_j be the number of occurrences of its negation, \bar{x}_j . The idea of the reduction is to exploit the feature of the oriented matching problem used in Example 2.1, namely that in a 3-cycle of first choices, at most one node can be in the set A of a valid matching. In our reduction, such cycles are constructed for the clauses, and all of their nodes can be matched only if some other node (corresponding to a true literal) is matched to one of them. Now we specify the construction formally.

The nodes in the constructed instance are partitioned into several sets, which are C (corresponding to clauses), T (corresponding to assigning the value *true* to variables), F (for value *false*), as well as E and S (some extra nodes for clean-up purposes). The set C is further partitioned into subsets $C_1 \dots C_m$, one for each clause, each of which contains three nodes, which correspond to the three literals in this clause. The sets T and F are partitioned into subsets $T_1 \dots T_k$ and $F_1 \dots F_k$, respectively, one each for each variable. A set T_j contains t_j nodes, and a set F_j contains f_j nodes. The set E contains $|T| + |F| = 3m$ nodes, and the set S contains m nodes. This makes $n = 10m$ nodes in total.

For the three nodes in each set C_i , let $\tau : C \rightarrow C$ map them to each other in a cycle. Let $\rho : T \cup F \rightarrow C$ be a bijection mapping each node in T_j to a node in C which corresponds to an occurrence of the literal x_j in the formula, and mapping the set F to the occurrences of negated variables. There is also a bijection $\sigma : E \rightarrow T \cup F$.

The preferences for all nodes are shown below. For a particular node a , the list goes from its most-preferred to its least-preferred nodes. A name of a set occurring in this list indicates that all the nodes in this set, except a

itself, in arbitrary order among each other, are preferred more than the later items on the list and less than the earlier ones.

- $a \in C_i$: $\tau(a)$; then $T \cup F \cup E \cup S$; then $C \setminus \{\tau(a)\}$
- $a \in T_j$: F_j ; then $\rho(a)$; then $(T \cup F \cup E \cup S) \setminus F_j$; then $C \setminus \{\rho(a)\}$
- $a \in F_j$: T_j ; then $\rho(a)$; then $(T \cup F \cup E \cup S) \setminus T_j$; then $C \setminus \{\rho(a)\}$
- $a \in E$: S ; then $\sigma(a)$; then $(T \cup F \cup E) \setminus \{\sigma(a)\}$; then C
- $a \in S$: $T \cup F \cup E \cup S$; then C

LEMMA 3.1. *If a 3SAT formula is satisfiable, then the corresponding instance constructed as above has a valid oriented matching μ of size $n/2$.*

Proof. We demonstrate a matching of size $n/2$, and then show that it satisfies the preference constraint. Choose a satisfying assignment for the formula, and then for each clause, select exactly one literal in it which is true in this assignment. Let the node corresponding to this literal be $c_i \in C_i$. In the matching $\mu : A \rightarrow B$ we include the following pairs:

- For each $c_i \in C_i$, include $(\rho^{-1}(c_i), c_i)$ and $(\tau(c_i), \tau(\tau(c_i)))$. This matches nodes in $T \cup F$ with their corresponding selected literals and the other literals to each other.
- For each $a \in E$ such that $\sigma(a) \in T \cup F$ is not matched yet, include $(a, \sigma(a))$.
- For each $a \in E$ such that $\sigma(a) \in T \cup F$ is matched to $\rho(\sigma(a)) \in C$, include the pair (a, s) for some $s \in S$ (different ones for different a 's)

This scheme matches all nodes. To make sure that there are enough nodes in S , note that exactly m (the number of clauses) nodes in $T \cup F$ are matched to their corresponding c_i 's. So m nodes in E are not matched to their σ -pair in $T \cup F$, and they can be exactly matched to the m nodes in S .

To verify that the preference constraint holds for this matching, we check it for nodes in each set. But first we note an important property that for each variable x_j , the set A cannot contain nodes from both T_j and F_j , since we only include these nodes for satisfied literals. So if the satisfying assignment of the formula sets a variable x_j to true, then only nodes from T_j can be included in the set A , and if the variable is set to false, then only nodes from F_j can be in A .

Any node $c \in C$ which is in A is matched to its first-choice node $\tau(c)$, so it does not prefer any other node in A . Any node $a \in T_j$ that is matched to $\rho(a) \in C$ prefers its match to any node except the ones in F_j . But since the nodes from F_j cannot appear in A (by the property above), a must prefer its match to any

node in A . A symmetric argument holds for nodes in F_j . A node $a \in E$ which is matched to either $\sigma(a)$ or to $s \in S$ may only prefer nodes in S to its match. But since no node from S appears in A , we conclude that the preference constraint holds. \square

LEMMA 3.2. *If there is a valid matching $\mu : A \rightarrow B$ of size $n/2$ in the constructed instance, then the original 3SAT formula is satisfiable.*

Proof. Since in each set C_i the first-choice preferences form a 3-cycle, at most one node, say a , from each C_i can be in A . Since all nodes participate in the matching, it means that at least two nodes from C_i must be in the set B . Since at most one of these can be matched to a , there must be at least one node $c_i \in C_i$ which is in a match (x, c_i) , where $x \notin C_i$. Now, observe that, just by counting, at least $2m \geq 2$ nodes from $T \cup F \cup E \cup S$ must be in A . So if $x \neq \rho^{-1}(c_i)$, then x must prefer some other node in A to c_i . This is because all nodes except C_i and $\rho^{-1}(c_i)$ prefer nodes in $T \cup F \cup E \cup S$ to node c_i . Therefore it must be that $x = \rho^{-1}(c_i)$.

By the way the preferences are set up, no two pairs (y, c_1) and (z, c_2) can be in the matching, where $y \in T_j$, $z \in F_j$, and $c_1, c_2 \in C$. This is because nodes in T_j and F_j prefer each other to any nodes in C . So if for each pair $(\rho^{-1}(c), c)$ that is in μ we set the literal corresponding to c to true, we get a consistent assignment that satisfies all the clauses. \square

This concludes the proof that the maximum T -matching problem is NP-hard. The same reduction shows that this problem is also APX-complete. This is because MAX 3SAT is APX-complete [10], and in our reduction, at least as many nodes remain unmatched in a valid matching as there are unsatisfied clauses in any truth assignment for the formula. Then the result follows by noting that the size of our oriented matching instance is bigger only by a constant factor than the size of the 3SAT formula (since $n = 10m$). This proves the following theorem.

THEOREM 3.1. *The decision version of the problem of finding a maximum-size preference-constrained oriented matching is NP-complete, and its optimization version is APX-complete.*

We now consider a different formulation of the problem. Whereas in our original formulation, any node of G could potentially be placed on either side of the oriented matching, in the new formulation the sides are prescribed as part of the input. More formally, the given set of nodes V consists of two disjoint subsets $V = S \cup T$, and the required preference-constrained matching has

to satisfy $A \subseteq S$ and $B \subseteq T$. We next show that this version of the problem is as hard to approximate as INDEPENDENT SET.

THEOREM 3.2. *The problem of finding a maximum-size preference constrained matching with prescribed sides is hard to approximate within a factor of $n^{1-\epsilon}$, for any $\epsilon > 0$, unless $NP=ZPP$. The result holds even for the case that G is a complete graph.*

Proof. We present a reduction from MAXIMUM INDEPENDENT SET, which is known to be hard to approximate within $n^{1-\epsilon}$, for any $\epsilon > 0$, unless any problem in NP can be solved in probabilistic polynomial time (i.e. $NP=ZPP$) [7]. Let $G = (V, E)$ be the given instance of independent set. For each vertex $v \in V$, we create nodes $s_v \in S$ and $t_v \in T$. Let $N_v \subseteq V$ be the set of neighbors of v in the original instance G . The preferences of node s_v can be expressed as

$$\{s_{v'} \mid v' \in N_v\} \succ t_v \succ \{s_{v'} \mid v' \in V \setminus N_v\} \cup \{t_{v'} \mid v' \neq v\},$$

with elements within sets ordered arbitrarily. Preferences of nodes in T are not relevant to the problem and can be arbitrary. We now show that the original instance G has an independent set of size k if and only if the derived instance admits a valid matching of size k , concluding the proof.

(\rightarrow) Let $U \subseteq V$ be an independent set in the original instance G . For every $v \in U$, add (s_v, t_v) to an oriented matching μ . Then $|\mu| = |U|$ and μ satisfies the preference constraint.

(\leftarrow) Let $\mu : A \rightarrow B$, with $A \subseteq S$ and $B \subseteq T$, be a valid matching in the derived instance. We let $U = \{v \mid s_v \in A\}$ and argue that it is an independent set in G . Indeed, if u and v are adjacent in G and both s_u and s_v are in A , then s_u would prefer s_v to whichever node $t_w \in T$ it is matched, violating the preference constraint. \square

4 A lower bound for coordination mechanisms

In this section we use Theorem 2.1 to correct an error in the proof of Theorem 5.2 in [1]. The setting studied by Azar, Jain, and Mirrokni [1] is as follows. First consider an instance of the unrelated machine scheduling problem, $R||C_{\max}$, with m machines and n jobs. Each job j takes p_{ij} amount of processing time if placed on machine i . Given a schedule that assigns all jobs to machines and orders the jobs assigned to each machine, the *completion time* C_j of a job j that is assigned to machine i is the sum of processing times on i of j and the jobs scheduled before j on machine i . We only consider non-preemptive schedules with no idle time between jobs. The optimization problem then is to

find a schedule that minimizes the maximum completion time, $C_{\max} = \max_j C_j$.

Within the framework of the above scheduling problem, the paper of Azar et al. studies the quality of different *coordination mechanisms*. In particular, each job in the scheduling instance is considered to be an independent selfish agent, whose goal is to minimize its own completion time. For this purpose, it is able to assign itself to any machine. The machines, on the other hand, have deterministic *ordering policies*, which are rules that determine, for any set of jobs placed on a machine, the order in which these jobs are scheduled. A coordination mechanism is defined by the set of ordering policies on the machines. A Nash Equilibrium in this setting is an assignment of jobs to machines such that no individual job can improve its own completion time by switching to a different machine. The *price of anarchy* for a particular coordination mechanism is the maximum, over all instances and all Nash Equilibria, of the ratio of the objective function C_{\max} in the Nash Equilibrium to the optimum value of C_{\max} for this instance.

The result considered here places a lower bound on the price of anarchy for *local policies* satisfying the *IIA property*. These restrictions concern the types of information that an ordering policy is allowed to use. A policy for machine i is *local* if it is only allowed to depend on IDs and full vectors of processing times (including processing times on other machines) for jobs placed on machine i , but not on any parameters of jobs placed on other machines. A policy satisfies the IIA property (independence of irrelevant alternatives) if the relative order of two jobs j and j' does not depend on the presence, absence, or any parameters of any third job j'' .

The following proof is based on the structure of the one in [1], but it corrects an error by iteratively using our result on the existence of large valid oriented matchings.

THEOREM 4.1. ([1], THEOREM 5.2) *The price of anarchy for all deterministic non-preemptive local policies satisfying the IIA property for $R||C_{\max}$ is at least $\Omega(\log m)$.*

Proof. Fix a set of m machines and a set of local ordering policies for them. We construct a set of jobs and a Nash Equilibrium for these jobs such that the cost of the optimal solution for the instance is 1, and the cost in the Nash Equilibrium is $\Omega(\log m)$.

We start with $\binom{m}{2}$ jobs, then select a subset of them to be part of the constructed instance, and discard the others. For each unordered pair of machines $\{i, i'\}$, create a job labeled by this pair, $j = (i, i') = (i', i)$. This job has processing time $p_{ij} = p_{i'j} = 1$ on machines i and i' , and $p_{i''j} = \infty$ on all other machines. Let G_1 be a

complete directed graph on m nodes that correspond to the machines. Define preferences of machines over each other using their ordering policies, such that $i' \succ_i i''$ whenever the policy of machine i places job (i, i') before job (i, i'') . For each index k starting from $k = 1$, let $\mu_k : A_k \rightarrow B_k$ be the maximum valid matching on G_k , and use it to define a set of jobs corresponding to its edges, $J_k = \{(i, \mu_k(i)) \mid i \in A_k\}$. The graph G_{k+1} is defined as a subgraph of G_k induced by the nodes in the subset A_k , with the same preferences as before. Let J_K be the last non-empty set of jobs found in this way, and note that the produced sets of jobs J_k are disjoint.

Theorem 2.1 implies that the size of each matching μ_k is at least a third of the number of nodes in G_k . This means that $|A_k| \geq \frac{m}{3^k}$ for k between 1 and K , and thus $K = \Omega(\log m)$. Our constructed scheduling instance consists of the m machines and the jobs in $\bigcup_k J_k$. We demonstrate an optimal solution to this instance with $C_{\max} = 1$ as well as a Nash Equilibrium with $C_{\max} = K$, proving the theorem. For each job $(i, i') \in J_k$, where $i' = \mu_k(i)$, the optimal solution places this job on machine $i' \in B_k$. In this way each job is placed on a machine where its processing time is 1. Also, each machine gets at most one job. To see this, note that all sets B_k are disjoint, and a machine $i' \in B_k$ can only get a job from the set J_k . Moreover, it can get only one such job, namely the one corresponding to its matched edge in μ_k .

A Nash Equilibrium is constructed by placing each job $(i, i') \in J_k$ on the machine $i \in A_k$. We note some properties of the sets involved and of the assignment. From the construction above it follows that $A_K \subset A_{K-1} \subset \dots \subset A_1$, and for any $k \geq 2$, $B_k \subset A_{k-1}$. The assignment of jobs is such that each machine i gets the set of jobs $\{(i, \mu_k(i)) \mid A_k \ni i\}$, and thus a load equal to $|\{A_k \mid i \in A_k\}|$, i.e. the number of sets A_k that i is in. Furthermore, all jobs in J_k have completion time equal to k . This follows from the way we defined the preferences based on the ordering policies and the fact that matchings μ_k are valid. In particular, a job $j = (i, \mu_1(i)) \in J_1$ will be the first in the ordering on machine i . This is because for any other job $j' = (i, \mu_k(i)) \in J_k$, $k > 1$, that is placed on machine i , we have $\mu_k(i) \in A_1$, and the preference constraint of μ_1 implies that i prefers $\mu_1(i)$ to $\mu_k(i)$. Inductively, the same argument shows that all jobs in J_2 are placed second (since they are placed on machines in $A_2 \subset A_1$ which already contain jobs from J_1), and so on. Thus, the maximum completion time of K is experienced by jobs in J_K .

To verify that this solution is a Nash equilibrium, we consider a particular job $j = (i, i') \in J_k$ that is currently on machine $i \in A_k$, and check whether it can

improve its completion time by switching. It would not switch to any machine $i'' \neq i'$, since $p_{i''j} = \infty$, so the only possible switch is to machine i' . Machine i' is in the set $B_k \subset A_{k-1}$, which means that it has a load of $k-1$. Now we claim that if job j switches to i' , then the ordering policy of i' will place it after all the $k-1$ jobs that are already on i' , and so its completion time would still be equal to k , providing no incentive for the switch. Consider any job $(i', i'') \in J_l$, with $\mu_l(i') = i''$ for some $l < k$, that is currently assigned to machine i' . Since $i \in A_l$, the preference constraint of the matching μ_l implies that $i'' \succ_{i'} i$, and correspondingly the job (i, i') would be placed after (i', i'') by the ordering policy of machine i' . \square

Acknowledgements

We thank Chien-Chung Huang and Peter Winkler for discussions of the problem.

References

- [1] Y. Azar, K. Jain, and V. Mirrokni. (Almost) optimal coordination mechanisms for unrelated machine scheduling. In *Proc. 19th ACM Symp. on Discrete Algorithms*, 2008.
- [2] J. Edmonds. Paths, trees, and flowers. *Canad. J. Math.*, 17:449–467, 1965.
- [3] L. Ford and D. Fulkerson. *Flows in Networks*. Princeton University Press, 1962.
- [4] D. Gale and L. Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 69(1):9–15, 1962.
- [5] D. Gale and M. Sotomayor. Some remarks on the stable matching problem. *Discrete Applied Mathematics*, 11(3):223–232, 1985.
- [6] D. Gusfield and R. Irving. *The Stable Marriage Problem*. The MIT Press, 1989.
- [7] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182:105–142, 1999.
- [8] R. W. Irving. An efficient algorithm for the “stable roommates” problem. *J. Algorithms*, 6(4):577–595, 1985.
- [9] D. Knuth. *Mariages stables et leurs relations avec d’autres problèmes combinatoires*. Les Presses de l’université de Montréal, 1976.
- [10] C. H. Papadimitriou and M. A. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.