Simultaneous Matchings: Hardness and Approximation ⋆

Martin Kutz [1]

Khaled Elbassioni
Max-Plank-Institut für Informatik, Saarbrücken, Germany.

Irit Katriel [2]
Brown University, Providence, RI, USA.

Meena Mahajan *,[2]
The Institute of Mathematical Sciences, Chennai 600 113, INDIA

Abstract

Given a bipartite graph $G = (X \dot\cup D, E \subseteq X \times D)$, an *$X$-perfect matching* is a matching in G that covers every node in X. In this paper we study the following generalisation of the X-perfect matching problem, which has applications in constraint programming: Given a bipartite graph as above and a collection $\mathcal{F} \subseteq 2^X$ of k subsets of X, find a subset $M \subseteq E$ of the edges such that for each $C \in \mathcal{F}$, the edge set $M \cap (C \times D)$ is a C-perfect matching in G (or report that no such set exists). We show that the decision problem is NP-complete and that the corresponding optimisation problem is in APX when $k = O(1)$ and even APX-complete already for $k = 2$. On the positive side, we show that a $2/(k+1)$-approximation can be found in $\mathrm{poly}(k, |X \cup D|)$ time. We show also that such an approximation M can be found in time $(k + \binom{k}{2}2^{k-2})\mathrm{poly}(|X \cup D|)$, with the further restriction that each vertex in D has degree at most 2 in M.

Key words: Matchings, perfect matchings, constraint programming, NP-completeness, optimisation, hardness of approximation.

1. Introduction

Matching is one of the most fundamental problems in algorithmic graph theory. The all-important notion of characterising feasibility / efficiency as polynomial-time computation came about in the context of the first efficient matching algorithm due to Edmonds [4]. Since then, an immense amount of research effort has been directed at understanding the various nuances and variants of this problem and at attacking special cases. For an overview of developments in matching theory and some recent algorithmic progress, see for instance [7,12].

In this paper we consider a generalisation of the bipartite matching problem. Suppose we are given a bipartite graph $G = (V, E)$ where the vertex set partition is $V = X \dot\cup D$ (so $E \subseteq X \times D$) and a collection $\mathcal{F} \subseteq 2^X$ of k *constraint sets*. A solution to the problem is a subset $M \subseteq E$ of the edges such that M is *simultaneously* a perfect matching for each constraint set in $\mathcal{F}$. More precisely, for each $C \in \mathcal{F}$, the edge set $M \cap (C \times D)$ has to be a C-perfect matching, i.e., a subgraph of G in which every vertex has degree at most 1 and every vertex of C has degree exactly 1. Also, analogous to maximum-weight matchings, we may relax the perfect matching condition and ask for a maximum-weight set M such that for each $C \in \mathcal{F}$, the edge set $M \cap (C \times D)$ is a matching in G.

Why consider this generalisation of matching? Apart from purely theoretical considerations suggesting that any variant of matching is worth exploring, there is a concrete important application in constraint programming where precisely this question arises. A *constraint program* consists of a set X of variables and a set D of values. Each variable $x \in X$ has a *domain* $D(x) \subseteq D$, i.e., a set of values it can take. In addition, there is a set of *constraints* that specify which combinations of assignments of values to variables are permitted.

An extensively studied constraint is the *AllDifferent* constraint (*AllDiff*) which specifies for a given set of variables that the values assigned to them must be pairwise distinct ([16], see also, e.g., [10,11,13,15,17]). An *AllDiff* constraint can be viewed as an X-perfect matching problem in the bipartite graph that has the set X of variables on one side, the set D of values on the other side, and an edge between each variable and each value in its domain. Typically, a constraint program contains several *AllDiff* constraints, defined over possibly overlapping variable sets. This setting corresponds to the generalisation we propose.

Formally, we consider the following problems:

SIM-W-MATCH (SIMULTANEOUS WEIGHTED MATCHINGS):
Input: a bipartite graph $G = (V, E)$ with $V = X \dot\cup D$ and $E \subseteq X \times D$, a weight $w(e)$ associated with each edge e in E, and a collection of constraint sets $\mathcal{F} \subseteq 2^X$.

⋆ A preliminary version appeared in the proceedings of ISAAC 2005, LNCS vol. 3827, pp. 106–115.
* Corresponding Author.
Email addresses: elbassio@mpi-sb.mpg.de (Khaled Elbassioni), irit@cs.brown.edu (Irit Katriel), meena@imsc.res.in (Meena Mahajan).
[1] Our friend and colleague Martin Kutz died in tragic circumstances while this manuscript was under review.
[2] This work was done while Irit Katriel was at the Max-Plank-Institut für Informatik, Saarbrücken, Germany, and while Meena Mahajan was visiting there.

Feasible Solution: a set $M \subseteq E$ satisfying $\forall C \in \mathcal{F} : M \cap (C \times D)$ is a matching. The weight of this solution is $\sum_{e \in M} w(e)$.
Output: (The weight of) a maximum-weight feasible solution.

SIM-W-PERF-MATCH (SIMULTANEOUS WEIGHTED PERFECT MATCHINGS):
Input: as above
Feasible Solution: as above but only saturating (perfect) matchings allowed, i.e., a set $M \subseteq E$ satisfying $\forall C \in \mathcal{F}$: "$M \cap (C \times D)$ is a C-perfect matching."
Output: (The weight of) a maximum-weight feasible solution or a flag indicating the absence of any feasible solution.

These are the optimisation / search versions [3]; in the decision versions, an additional weight W is given as input and the answer is 'yes' if there is a feasible solution of weight at least W. When all edge weights are 1, the corresponding problems are denoted by SIM-MATCH and SIM-P-MATCH respectively. In this case, the decision version of SIM-P-MATCH does not need any additional parameter W.

We use the following notation: $n = |X|$, $d = |D|$, $m = |E|$, $k = |\mathcal{F}|$, $t = \max\{|C| : C \in \mathcal{F}\}$. We also assume, without loss of generality, that $X = \cup_{C \in \mathcal{F}} C$.

At first sight these problems do not appear much more difficult than bipartite matching, at least when the number of constraint sets is a constant. It seems quite plausible that a modification of the Hungarian method [9] should solve this problem. However, we show in Section 2 that this is not the case; even when $k = 2$, SIM-P-MATCH is NP-hard. We also show that it remains NP-hard even if the graph is complete bipartite (i.e., $E = X \times D$) and d and t are constants; of course, in this case, k must be unbounded. Furthermore, SIM-MATCH is APX-hard, even for $k = 2$. On the positive side, SIM-W-MATCH is in APX for every constant k. These results are shown in Section 3. Finally, in Section 4 we examine the SIM-P-MATCH polytope and observe that it can have vertices that are not even half-integral.

2. NP-completeness of SIM-P-MATCH for $k \geq 2$

The main result of this section is the following.

Theorem 1 *Determining feasibility of an instance of* SIM-P-MATCH *with k constraint sets is NP-complete for every single parameter $k \geq 2$.*

Proof: Membership in NP is straightforward. We establish NP-hardness of SIM-P-MATCH for $k = 2$ (it then follows trivially for each $k > 2$). The proof is by reduction from SET-PACKING.

SET-PACKING :
Instance: A universe U; a collection $\mathcal{C} = \{S_1, S_2, \ldots, S_p\}$ of subsets of U.
Decision problem: Given an integer $\ell \leq p$, is there a collection $\mathcal{C}' \subseteq \mathcal{C}$ of at least ℓ pairwise disjoint sets?
Optimisation problem: Find a collection $\mathcal{C}' \subseteq \mathcal{C}$ of pairwise disjoint subsets such that $|C'|$, the number of chosen subsets, is maximised.

[3] Note that due to the weights, an optimal solution to the former might not saturate all sets even if such a perfect assignment exists. Thus SIM-W-PERF-MATCH is not a special case of SIM-W-MATCH.

k-SET-PACKING: The restriction where every set in $\mathcal{C}$ contains at most k elements.

SET-PACKING(r): The restriction where every element of U appears in at most r sets from $\mathcal{C}$.

It is known that SET-PACKING is NP-hard, and so is 3-SET-PACKING(2), the special case where the size of each set is bounded by 3 and each element occurs in at most 2 sets. See, for instance, [2].

We present the reduction from SET-PACKING to SIM-P-MATCH (with $k = 2$) in detail here because it will later serve for an APX result, too.

View SIM-P-MATCH as a question of assigning values (from D) to variables in X (as in the constraint programming application described in Section 1). Formally, there is a distinct node (value) u_S in D for each set $S \in \mathcal{C}$, and there is also a distinct value v_a in D for each element $a \in U$. In X, we place ℓ variables $x_1, \ldots, x_\ell$, which belong to both constraint sets X_1 and X_2. These variables are connected to all the *set* values, but not to any of the *element* values. Since they are contained in X_1 and X_2, any simultaneous perfect matching must match these variables to exactly ℓ distinct set values. This corrresponds to picking ℓ of the p sets, and we must now ensure that these chosen ℓ sets are indeed disjoint. We do this by placing, between the value nodes of the form u_S and v_a, gadgets that encode the composition of the sets.

Consider the trivial situation with only singleton sets in $\mathcal{C}$. Then we could simply identify the set node with the corresponding element node. This results in each x_i being connected to each value that occurs as such a singleton. Then a "packing" of ℓ singleton sets in U would obviously give an assignment of ℓ values to the x_i's in this complete bipartite graph, and vice versa.

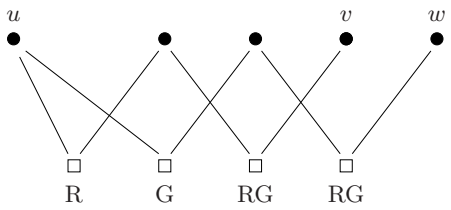

Fig. 1. The basic gadget for the reduction.

The difficult part is to build gadgets that force disjointness even when the sets are larger. Choosing a non-singleton set corresponds to a single variable occupying more than one value from U. Consider the case when the set u has two elements, v and w. We construct the configuration in Figure 1. We have five values on the upper side and four variables on the lower, marked with letters 'R' (red) and 'G' (green) to indicate that they belong to the constraint sets X_1 and X_2, respectively (red-green colour indicating membership in both sets). The leftmost value u is the set value described before; the rightmost two values v, w are the element values also described before. The middle two values, and all four variables, are new and are added solely to encode this set. If the set value is grabbed by an x_i variable (or any other red-green variable) outside the figure, then the two left new variables will be forced to claim the two values to their right. In turn, the remaining two variables will have to pick the values marked v and w. In other words, if a red-green variable claims the *input* value u on the left, it effectively occupies the two *output* values v and w, too. Conversely, if the value u is not required

elsewhere, the four variables can all make their left-slanted connections and leave v and w untouched.

We can concatenate several such 4-variable gadgets to obtain a larger amplification. If we identify an output value v of one gadget with the input u' of another one, as shown in Figure 2, we get the effect that occupying only the input u from outside this configuration, forces the gadget variables to claim the three output values w, v', and w' that could otherwise stay untouched. (Indeed, we only get three such values and not four because the connecting value v counts no longer as an output.)

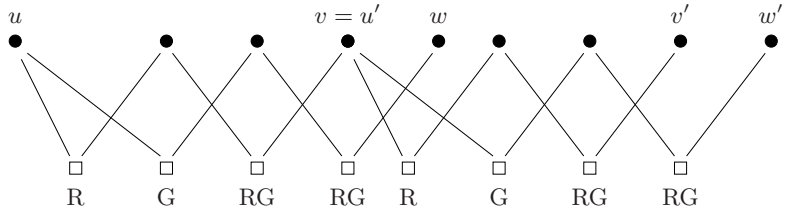

Fig. 2. Merging two 1:2-gadgets into one 1:3-gadget.

For a q-element set $S = \{v_1, \ldots, v_q\} \in \mathcal{C}$, we concatenate $q - 1$ gadgets and make $v_1, \ldots, v_q \in U$ their resulting output values.

The resulting configuration obviously has the desired behaviour. We can assign values to all variables without violating the red and green constraints if and only if we can pack ℓ sets from $\mathcal{C}$ into U. □

A complete example for the NP-hardness reduction is shown in Figure 3.

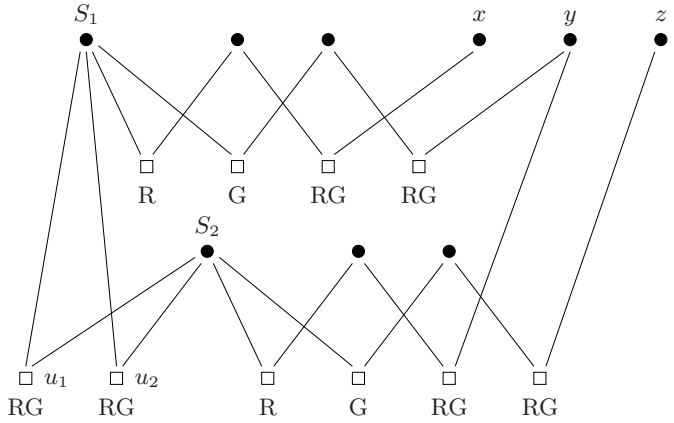

Fig. 3. The SIM-P-MATCH instance obtained from the unsolvable 3-SET-PACKING(2) instance $S_1 = \{x, y\}$, $S_2 = \{y, z\}$ and $\ell = 2$.

Complete bipartite graphs with $d = 3$, $t = 2$. Theorem 1 states that it is NP-hard to solve instances of SIM-P-MATCH with two constraint sets X_1 and X_2. However, if the graph is a complete bipartite graph, it is straightforward to determine whether a solution exists and to find it if so: First match the vertices of X_1 with any set of distinct vertices in D. Then it remains to match $X_2 \setminus X_1$ with vertices that were not matched with $X_1 \cap X_2$. Since the graph is complete bipartite, the existence of a solution is determined solely by the sizes of X_1, X_2, $X_1 \cap X_2$ and D.

It is therefore natural to ask whether it is always possible to solve SIM-P-MATCH on a complete bipartite graph. With a little bit of thought and inspection, one can come up with similar feasibility conditions for $k = 3, 4$. What about arbitrary k? It turns out that the problem is NP-hard if the number of constraint sets is not bounded, even if each constraint set has cardinality 2 and $d = |D| = 3$. The proof is by a reduction from 3-VERTEX-COLORING, which is known to be NP-complete (see for instance [5]).

3-VERTEX-COLORING :
Instance: An undirected graph $G = (V, E)$.
Decision problem: Is there a way of colouring the vertices of G, using at most 3 distinct colours, such that no two adjacent vertices get the same colour?

Proposition 1 SIM-P-MATCH *is NP-hard even when* $d = 3$, $t = 2$, *and the underlying graph is complete bipartite.*

Proof: Let $G = (V, A)$ be an instance of 3-VERTEX-COLORING. We construct a corresponding instance of SIM-P-MATCH as follows: let $X = V$, $D = \{1, 2, 3\}$, $E = X \times D$, and $\mathcal{F} = A$. It is straightforward to see that any feasible solution to this instance of SIM-P-MATCH is a 3-colouring of V with no monochromatic edge, and vice versa. □

3. APX-completeness

We now examine the approximability of SIM-W-MATCH.

3.1. *Membership in APX for constant k*

APX is the class of optimisation problems which have polynomial-time constant-factor approximation algorithms. That is, a maximisation problem Π is in APX if there is a constant $0 < \alpha \leq 1$ and a polynomial-time algorithm A such that for every instance x of Π we have $\alpha \cdot \mathrm{Opt}(\Pi, x) \leq A(x) \leq \mathrm{Opt}(\Pi, x)$, where $A(x)$ is the value of the output of the algorithm on input x, and $\mathrm{Opt}(\Pi, x)$ is the value of the optimum solution. Clearly, the larger the *approximation factor* α, the better the quality of approximation. For more details, see any text on approximation algorithms, such as [6,18].

Consider the following naive polynomial-time approximation algorithm for SIM-W-MATCH: Find a maximum weight matching for each constraint set $C \in \mathcal{F}$ independently and return the heaviest matching found. Clearly, an optimal solution is at most k times larger, which gives an approximation ratio of $1/k$. Hence we have:

Proposition 2 *An instance of* SIM-W-MATCH *with k constraint sets can be approximated in polynomial time within a factor of* $1/k$.

Corollary 1 *For any constant k,* SIM-W-MATCH *with k constraint sets is in APX.*

3.2. *An improved approximation factor*

We can slightly improve the $1/k$ factor by considering more than one set $X_i \in \mathcal{F}$ at a time. Fix a maximum-weight feasible solution M with weight Opt. For any set $S \subseteq X$, let $f(S)$ denote the weight of the maximum weight simultaneous matching in the graph induced by $S \cup D$, and let $g(S)$ be the weight of the edges in $M \cap (S \times D)$. Clearly, for

each S we have $f(S) \geq g(S)$ and further, each feasible solution on $S \cup D$ is also a solution on G, so that $\mathrm{Opt} \geq f(S)$.

First consider the case with two constraint sets X_1, X_2. We can compute $f(X_1)$ and $f(X_2)$ independently, each as an ordinary maximum-matching problem, and we can also evaluate the symmetric difference $f(X_1 \oplus X_2) := f(X_1 \setminus X_2) + f(X_2 \setminus X_1)$ as the union of two independent maximum-matching problems. Altogether we get

$$f(X_1) + f(X_2) + f(X_1 \oplus X_2) \geq g(X_1) + g(X_2) + g(X_1 \oplus X_2) = 2g(X_1 \cup X_2) = 2\,\mathrm{Opt}.$$

By averaging, the largest of the three terms on the left is at least $2/3 \cdot \mathrm{Opt}$.

For $k > 2$, we can generalise the above method as follows. Define $\hat{X}_i := X_i \setminus \bigcup_{h \neq i} X_h$, then $X_1 \oplus \ldots \oplus X_k = \bigcup_{i=1}^{k} \hat{X}_i$. Then we can efficiently compute $f(X_1 \oplus \ldots \oplus X_k) = \sum_{i=1}^{k} f(\hat{X}_i)$ as the union of k independent maximum matching problems. Thus our approximation in this case will be the maximum among the individual values of $f(X_1)$, …, $f(X_k)$ and the value of $f(X_1 \oplus \cdots \oplus X_k)$. The quality of this approximation can be bounded as follows.

$$\begin{aligned}
\max\{f(X_1), \ldots, f(X_k), f(X_1 \oplus \cdots \oplus X_k)\} &\geq \frac{1}{k+1}\left(\sum_{i=}^{k} f(X_i) + f(X_1 \oplus \cdots \oplus X_k)\right) \\
&\geq \frac{1}{k+1}\left(\sum_{i=}^{k} g(X_i) + g(X_1 \oplus \cdots \oplus X_k)\right) \\
&\geq \frac{2}{k+1}\, g(X) = \frac{2}{k+1}\,\mathrm{Opt},
\end{aligned}$$

where the last inequality follows from the fact that each element in X is counted at least twice in the sets $X_1, \ldots, X_k, X_1 \oplus \cdots \oplus X_k$. Thus we obtain the following general result:

Theorem 2 SIM-W-MATCH *can be approximated within a factor of* $2/(k+1)$ *by an algorithm that runs in* $k\mathrm{poly}(n, m, d)$ *time.*

Looking more carefully at the argument above, it may seem that the selection of the family of sets $X_1, \ldots, X_k, X_1 \oplus \ldots \oplus X_k$ was completely arbitrary. In fact, for any family $\mathcal{F}' \subseteq 2^X$ of subsets of X, we can similarly define the following general approximation procedure:

Procedure SIM-W-MATCH-APPROX$(G, \mathcal{F}, \mathcal{F}')$:
Input: a bipartite graph $G = (V, E)$ with $V = X \,\dot\cup\, D$ and $E \subseteq X \times D$, a weight $w(e)$ associated with each edge in E, a collection of constraint sets $\mathcal{F} \subseteq 2^X$, and family $\mathcal{F}' \subseteq 2^X$ of subsets of X.
Output: a set $M \subseteq E$ satisfying $\forall C \in \mathcal{F} : M \cap (C \times D)$ is a matching.
Computation: Return $\max\{f(S) \ : \ S \in \mathcal{F}'\}$.

Clearly, in order to be able to implement this procedure, we require that the family $\mathcal{F}'$ is selected such that $f(S)$ can be computed in polynomial time for each $S \in \mathcal{F}'$. A sufficient condition (which is also necessary in view of Theorem 1, assuming $P \neq NP$) for this is that, for every $S \in \mathcal{F}'$:

(*) there exists an $\ell \in [k]$, a set of distinct indices $p_1, \ldots, p_\ell \in [k]$, and an onto mapping $\phi : S \mapsto \{p_1, \ldots, p_\ell\}$, such that

$$(C1)\ \forall x \in S : \ x \in X_{\phi(x)},$$
$$(C2)\ \forall x, y, i : \ x, y \in S \cap X_i \Longrightarrow \phi(x) = \phi(y).$$

For instance, if $k = 3$ and we take $S = (X_1 \setminus X_2) \cup (X_2 \setminus (X_1 \cup X_3))$, then (*) is satisfied for S, by taking $\ell = 2$, $\phi(x) = 1$ for $x \in S \cap X_1$, and $\phi(x) = 2$ for $x \in S \cap X_2$. However, for $S = (X_1 \setminus X_2) \cup (X_2 \setminus (X_1 \cup X_3)) \cup (X_1 \cap X_2 \cap X_3)$, no such mapping exists, for if we take $x \in S \cap (X_1 \setminus X_2)$, $y \in S \cap (X_2 \setminus (X_1 \cup X_3))$ and $z \in S \cap (X_1 \cap X_2 \cap X_3)$, then $(C1)$ implies that $\phi(x) = 1$, $\phi(y) = 2$ and $\phi(z) \in \{1, 2, 3\}$, while $(C2)$ implies that $\phi(x) = \phi(z) = \phi(y)$, a contradiction.

If condition (*) is satisfied for $S \in \mathcal{F}'$, then a maximum weight simultaneous matching on S can be evaluated as the disjoint union of maximum weight matchings on the sets $S \cap X_{p_i}$, computed independently for each $i \in [\ell]$:

$$f(S) = \sum_{i=1}^{\ell} f(S \cap X_{p_i}), \tag{1}$$

(To see (1), let M_S be a maximum weight matching on S. Note that by $(C1)$ and $(C2)$, the sets $S \cap X_{p_i}$ are pairwise disjoint, and thus $S = \dot{\cup}_{i\in[\ell]}(S \cap X_{p_i})$ and $w(M_S) = \sum_{i\in[\ell]} w(M_S \cap (X_{p_i} \times D)) \leq \sum_{i\in[\ell]} f(S \cap X_{p_i})$. On the other hand, if M_i is a matching on $S \cap X_{p_i}$ then the union $\dot{\cup}_{i\in[\ell]} M_i$ is a simultaneous matching on S, for otherwise there exist two variables $x, y \in X_i$ for some $i \in [k]$, such that x, y are matched, respectively by $M_{\phi(x)}$ and $M_{\phi(y)}$, to the same value in D. But then $(C2)$ would imply that $\phi(x) = \phi(y)$, in contradiction with the fact that x and y are matched to the same value.)

We shall call any subset of X satisfying (*) a *feasible* set. For instance, in the proof of Theorem 2, the set $\mathcal{F}' = \{X_1, \cdots, X_k, X_1 \oplus \cdots \oplus X_k\}$ forms such a feasible family.

Now it becomes natural to ask the following question: Is it possible to choose a family of feasible subsets of X, different from the one chosen in Theorem 2, for which the application of the above approximation procedure gives a better approximation factor than $2/(k+1)$? Let us first consider an example.

Example 1 *For $i = 1, \ldots, k$, let $\hat{X}_i$ be as defined above and consider the pairs $S_{ij} = \hat{X}_i \cup (X_j \setminus X_i)$, for $i \neq j$. Then the family $\mathcal{F}' = \{X_i : i \in [k]\} \cup \{S_{ij} : 1 \leq i < j \leq k\}$ is feasible. A careful choice of coefficients yields*

$$\sum_i f(X_i) + \frac{1}{2(k-1)} \sum_{i\neq j} f(S_{ij}) \geq \sum_i g(X_i) + \frac{1}{2(k-1)} \sum_{i\neq j} g(S_{ij}) \geq 2g(X) = 2\,Opt$$

and again by averaging, at least one of the $f(X_i)$ or $f(S_{ij})$ is at least $4/(3k) \cdot Opt$. □

The fact that the approximation factor obtained in Example 1 is less than $2/(k+1)$, for $k > 2$, does not come as a coincidence. Perhaps surprisingly, we shall show below that no matter what feasible family of sets we pick, even if we allow exponentially large families, the maximum approximation ratio achievable by procedure SIM-W-MATCH-APPROX is $2/(k+1)$.

Theorem 3 *For every $k \geq 2$, and every feasible family $\mathcal{F}' \subseteq 2^X$, there exists an infinite family of instances $(G, \mathcal{F})$ of* SIM-W-MATCH *on which the approximation ratio achieved by procedure* SIM-W-MATCH-APPROX$(G, \mathcal{F}, \mathcal{F}')$ *is arbitrarily close to $2/(k+1)$.*

Let $\mathcal{F}'$ be a family of feasible sets (i.e. those satisfying (*)). Then, for every choice of non-negative weights ξ_S, $S \in \mathcal{F}'$, we have

$$\sum_{S\in\mathcal{F}'} \xi_S f(S) \geq \sum_{S\in\mathcal{F}'} \xi_S g(S) \geq F_\xi \cdot \mathrm{Opt}$$

where $F_\xi = \min_{x\in X} \sum_{S:x\in S} \xi_S$; the last inequality holds because each edge (x, d) of the optimal solution M is counted with a total weight of $\sum_{S:x\in S} \xi_S$. By averaging, the

largest term $f(S)$ is at least $F_\xi \cdot \mathrm{Opt}/T_\xi$, where $T_\xi = \sum \xi_S$ is the total weight. (In proving Proposition 2, the chosen S's were precisely the X_i's, with weight 1 each, so $F_\xi = 1$ and $T_\xi = k$. In Example 1, the chosen S's were the X_i's and the S_{ij}'s, so $F_\xi = 2$ and $T_\xi = 3k/2$.)

Clearly, we lose nothing by considering only *maximal* subsets S for which (*) is satisfied. (The sets S_{ij} in Example 1 were maximal in this sense, while the sets $\hat{X}_i \cup \hat{X}_j$ are not.) Below, we consider only maximal subsets. To prove Theorem 3, we identify the family of all maximal feasible subsets of X and show that under the best possible selection of weights ξ_S, the approximation ratio F_ξ/T_ξ is $2/(k+1)$.

Let $\mathcal{R}$ denote the collection of all possible maximal feasible sets, and $\mathcal{C}$ denote the family of maximal subsets of elements of X, that have the same classification with respect to containment in either of each X_i or $X \setminus X_i$, for $i = 1, \ldots, k$. That is, every $C \in \mathcal{C}$ is identified with a vector $(v_1, v_2, \ldots, v_k) \in \{0,1\}^k \setminus 0^k$, such that $C = \bigcap_{i:v_i=1} X_i \setminus \bigcup_{i:v_i=0} X_i$. Clearly, $\beta = |\mathcal{C}| = 2^k - 1$. Below, we shall characterise the sets in $\mathcal{R}$ and observe that $\alpha = |\mathcal{R}|$ is $O((2k)^k)$. Also, we note that, due to maximality, for each $R \in \mathcal{R}$ and each $C \in \mathcal{C}$, C is either contained in or disjoint from R. Note that we consider maximality under the "general position" assumption:

(A) for any $(v_1, v_2, \ldots, v_k) \in \{0,1\}^k \setminus 0^k$, the set $C = \bigcap_{i:v_i=1} X_i \setminus \bigcup_{i:v_i=0} X_i$ is non-empty.

(Clearly, making such an assumption only makes the problem harder.) We now wish to compute, for each $R \in \mathcal{R}$, a weight $\xi_R \geq 0$ such that the corresponding ratio F_ξ/T_ξ is maximised. Define an $\alpha \times \beta$ 0-1 matrix $A = (a_{R,C})_{R,C}$ where $a_{R,C} = 1$ iff $C \subseteq R$. Consider the following pair of primal-dual linear programs:

$$\begin{array}{lll} \lambda_P^*(k) = \max\ \lambda & & \lambda_D^*(k) = \min\ \lambda \\ \quad s.t. \quad A^T\xi \geq \lambda \mathbf{e}_\beta & & \quad s.t. \quad Az \leq \lambda \mathbf{e}_\alpha \\ \quad\quad \mathbf{e}_\alpha^T \xi = 1 & & \quad\quad \mathbf{e}_\beta^T z = 1 \\ \quad\quad \xi \geq \mathbf{0} & & \quad\quad z \geq \mathbf{0} \end{array} \tag{2}$$

over $\xi \in \mathbb{R}^\alpha$ and $z \in \mathbb{R}^\beta$, where $\mathbf{e}_\alpha$ and $\mathbf{e}_\beta$ are the vectors of all ones of dimensions α and β respectively. A feasible solution ξ to the primal assigns weights (normalised so that $T_\xi = 1$) to each $R \in \mathcal{R}$ achieving an approximation factor given by the value of the corresponding objective function. We show that both the primal and the dual have feasible solutions with objective value $2/(k+1)$.

Theorem 4 *For any integer $k \geq 1$, we have $\lambda_P^*(k) = \lambda_D^*(k) = \frac{2}{k+1}$.*

We shall strengthen Theorem 2 as follows. For any positive integer δ, let us call a simultaneous matching M δ-bounded if every value $v \in D$ has degree at most δ in M. Our proof of Theorem 4 will also imply the following.

Theorem 5 *A 2-bounded simultaneous matching within a factor of $2/(k+1)$ of the optimal simultaneous matching can be found by* SIM-W-MATCH-APPROX *in deterministic time $(k + \binom{k}{2}2^{k-2})\mathrm{poly}(n, m, d)$.*

3.3. *Characterizing maximal feasible sets and the proof of Theorems 3, 4 and 5*

Let $S \subseteq X$ be a maximal feasible set. Then by (*), S is completely determined by the selection of ℓ, the set of indices $P = \{p_1, \ldots, p_\ell\}$, and the onto mapping $\phi : S \mapsto P$. Furthermore, conditions $(C1)$ and $(C2)$ of (*), together with the maximality of S, imply that (a) all the elements of S in a single set X_i will be mapped to the same index in P, i.e. we can think of ϕ as mapping the sets rather than the elements, and (b) all the elements in $S \cap X_{p_i} \setminus (\cup_{j \neq i} X_{p_j})$ are mapped to p_i, for $i = 1, \ldots, \ell$. In particular, the family of maximal feasible subsets of X is in one-to-one correspondence with the set $\mathcal{R} = \mathcal{R}(k)$ of all possible sequences obtained from the elements of the set $[k]$ in the following way:

(i) Select a subset $P = \{p_1, \ldots, p_\ell\} \subseteq [k]$ of size ℓ, where $\ell \in [k]$. The elements of P are called the *parent* elements (and correspond to the sets $X_{p_1}, \ldots, X_{p_\ell}$). For a given ℓ, the number of ways to do this is $\binom{k}{\ell}$.

(ii) Assign each of the remaining $k - \ell$ elements of $[k]$ to exactly one parent from P (this corresponds to assigning the elements of each of the remaining sets X_i, $i \notin P$, to one of the sets X_{p_j}, $j \in P$); the number of ways to do this is $\ell^{k-\ell}$.

Thus,

$$|\mathcal{R}(k)| = \sum_{\ell=1}^{k} \binom{k}{\ell} \ell^{k-\ell}.$$

Note that every element $R \in \mathcal{R}(k)$ can be identified with a 2ℓ-tuple $(p_1, \ldots, p_\ell, B_1, \ldots, B_\ell)$ where $\ell \in [k]$ and $\{p_1, \ldots, p_\ell\}, B_1, \ldots, B_\ell \subseteq [k]$, such that $\mathcal{P}_R = \{\{p_1\} \cup B_1, \ldots, \{p_\ell\} \cup B_\ell\}$ forms a partition of $[k]$ into ℓ parts (blocks). For example, if $k = 3$ and $S = (X_1 \setminus X_2) \cup (X_2 \setminus (X_1 \cup X_3))$ (which is a maximal feasible set), then the corresponding $R \in \mathcal{R}(k)$ is $R = (1, 2, \{3\}, \emptyset)$.

Let $\mathcal{C} = \mathcal{C}(k) = 2^{[k]} \setminus \{\emptyset\}$ be the set of all possible non-empty subsets of $[k]$. We construct a matrix $A(k) = (a_{R,C})_{R,C} \in \{0,1\}^{\mathcal{R}(k) \times \mathcal{C}(k)}$ whose rows and columns are indexed by the elements of $\mathcal{R}(k)$ and $\mathcal{C}(k)$, respectively. For $R = (p_1, \ldots, p_\ell, B_1, \ldots, B_\ell) \in \mathcal{R}(k)$ and $C \in \mathcal{C}(k)$, we let $a_{R,C} = 1$ if and only if the set C is contained in a single part of the partition defined by R, i.e. $\{p_j\} \subseteq C \subseteq \{p_j\} \cup B_j$, for some $j \in [\ell]$. Denote by $\alpha = |\mathcal{R}|$, $\beta = |\mathcal{C}|$ and $A = A(k)$, and consider the pair of primal-dual linear programs given by (2) over $\xi \in \mathbb{R}^\alpha$ and $z \in \mathbb{R}^\beta$.

Lemma 1 *For $k \geq 2$, let $\xi_1^*, \ldots, \xi_\alpha^*$ be an optimal solution for the primal linear program (2), with objective value $\lambda_P^*(k)$. Then procedure* SIM-W-MATCH-APPROX$(G, \mathcal{F}, \mathcal{F}')$*, where $\mathcal{F}' = \{R \in \mathcal{R}(k) : \xi_R^* > 0\}$, is a $\lambda_P^*(k)$-approximation algorithm for* SIM-W-MATCH.

Proof: Given an instance $(X, D, E, \mathcal{F}, w)$ of SIM-W-MATCH, we construct for each $R \in \mathcal{R}(k)$ with $\xi_R^* > 0$, a corresponding family $\mathcal{F}(R)$ as follows. Assume as before that $\mathcal{F} = \{X_1, \ldots, X_k\}$. For each set $X_{p_i} \in \mathcal{F}$ such that $p_i \in [k]$ is a parent in R, the corresponding set $X_{p_i}(R) \in \mathcal{F}(R)$ consists of exactly those elements of X_{p_i} that do not belong to any set X' which is another parent or is assigned to another parent $p_j \neq p_i$. More precisely, if $R = (p_1, \ldots, p_\ell, B_1, \ldots, B_\ell)$, then

$$X_{p_i}(R) = X_{p_i} \setminus \left(\bigcup_{j \in [k] \setminus (\{p_i\} \cup B_i)} X_j \right),$$

and we let $\mathcal{F}(R) = \{X_{p_i}(R) \ : \ i = 1, \ldots, \ell\}$. For example, if $k = 8$, $\mathcal{F} = \{X_1, \ldots, X_8\}$, and $R = (1, 2, 3, \{4, 5\}, \{6\}, \{7, 8\})$, then $X_1(R) = X_1 \setminus (X_2 \cup X_3 \cup X_6 \cup X_7 \cup X_8)$, $X_2(R) =$

$X_2 \setminus (X_1 \cup X_3 \cup X_4 \cup X_5 \cup X_7 \cup X_8)$, and $X_3(R) = X_3 \setminus (X_1 \cup X_2 \cup X_4 \cup X_5 \cup X_6)$.

Let $\cup_{B \in \mathcal{F}(R)} B$ be denoted X_R. The instance $(X_R, D, E_R, \mathcal{F}(R), w)$ where $E_R = E \cap (X_R \times D)$ is easy to solve optimally since the sets in $\mathcal{F}(R)$ are all feasible (for each $B \in \mathcal{F}(R)$, obtain a maximum matching in $(B \cup D, E_R)$, and simply put these maximum matchings together). Let the total weight of this optimal solution be $f(R)$. Note that this solution is also a feasible solution for the original instance, though not necessarily optimal since it does not match anything outside X_R.

Now any solution to the input instance, and in particular an optimal solution, is partitioned by the k sets of $\mathcal{F}$ into at most $2^k - 1$ parts. These parts are in one-to-one correspondence with the elements of $\mathcal{C}(k)$. Fix an optimal solution M. For any $R \in \mathcal{R}$, let $g(R)$ denote the total weight of edges of M in $X_R \times D$ (i.e. the total weight of variables in X_R that are assigned values in M). Then for every R, we have $g(R) \leq f(R) \leq \text{Opt}$. Moreover, $g(R) = \sum_{C \in \mathcal{C}(k)} a_{R,C} g(C)$ by linearity of $g(\cdot)$, where we denote by $g(C)$, for $C \in \mathcal{C}(k)$, the weight of edges incident to $\cap_{i \in C} X_i \setminus \cup_{i \notin C} X_i$ in the optimal solution M. Lemma 1 then follows from the following claim.

Claim 1 $\max_{R \in \mathcal{R}(k)} f(R) \geq \lambda_P^*(k)\, Opt.$

Indeed, if for every $R \in \mathcal{R}$ we have $f(R) < \lambda_P^*(k)\text{Opt}$, then

$$\begin{aligned}\sum_R \xi_R^* g(R) &\leq \sum_R \xi_R^* f(R) \\ &< \sum_R \xi_R^* \lambda_P^*(k)\text{Opt} \\ &= \lambda_P^*(k)\text{Opt} \quad (\text{because } \mathbf{e}_\alpha^T \xi^* = 1).\end{aligned}$$

On the other hand,

$$\begin{aligned}\sum_R \xi_R^* g(R) &= \sum_R \xi_R^* \sum_C a_{R,C} g(C) \\ &= \sum_C g(C) \sum_R \xi_R^* a_{R,C} \\ &\geq \sum_C g(C) \lambda_P^*(k) \quad (\text{because } A^T \xi^* \geq \lambda_P^*(k) \mathbf{e}_\beta) \\ &= \lambda_P^*(k)\text{Opt},\end{aligned}$$

a contradiction. □

Proof of Theorem 3. Let $z^* = (z_1^*, \ldots, z_\beta^*)$ be a rational optimal solution for the dual linear program (2), and let L be an arbitrary integer such that $z' = Lz^* \in \mathbb{Z}_+^\beta$ is an integral vector. The instance is given by a set X of size $\sum_{C \in \mathcal{C}} z'_C + |\{C \in \mathcal{C} : z_C = 0\}|$, and a set D of size $\sum_{C \in \mathcal{C}} z'_C + 1$. The family $\mathcal{F} = \{X_1, \ldots, X_k\}$ will be defined by constructing the set of variables in the different parts corresponding to the sets $C \in \mathcal{C}$. Specifically, for every $C \in \mathcal{C}$, we include z'_C variables in the set $\cap_{i \in C} X_i \setminus \cup_{i \notin C} X_i$ if $z'_C > 0$, while we include only one variable if $z'_C = 0$. All the variables corresponding to $z'_C = 0$ are connected to the same value v_0 in D; all the other variables are connected to distinct values, different from v_0, so that they form a perfect matching with these values. Note that the optimum solution to the simultaneous matching problem has value at least $|D| = \sum_{C \in \mathcal{C}} z'_C + 1 = L + 1$. On the other hand, if we let, as before, $\mathcal{R}$ denote the family

of all maximal feasible sets, then the solution returned by the approximation procedure is at most

$$\begin{aligned}\max\{\sum_{C\in\mathcal{C}} a_{R,C}z'_C \ : \ R\in\mathcal{R}\}+k &= L\cdot\max\{\sum_{C\in\mathcal{C}} a_{R,C}z^*_C \ : \ R\in\mathcal{R}\}+k\\ &= L\cdot\max\{\xi^T Az^* : \mathbf{e}_\alpha^T\xi=1,\ \xi\geq\mathbf{0}\}+k\\ &\leq L\cdot\lambda^*_P(k)+k,\end{aligned}$$

where the last inequality follows from the feasibility condition $Az^* \leq \lambda^*_D(k)\mathbf{e}_\alpha$, which implies $\xi^T Az^* \leq \lambda^*_D(k)\xi^T\mathbf{e}_\alpha = \lambda^*_P(k)$, for any ξ such that $\mathbf{e}_\alpha^T\xi = 1$ and $\xi \geq \mathbf{0}$. The ratio of the approximation to the optimal solution is thus at most $\frac{L\lambda^*_P(k)+k}{L+1}$. This ratio tends to $\lambda^*_P(k)$ for $L \gg k$, and hence, Theorem 3 follows as a consequence of Theorem 4. □

To prove Theorem 4, it is enough to establish a primal-dual pair of solutions for (2) which achieve the same objective value of $2/(k+1)$. It is not difficult to see that, for $R = (p_1,\ldots,p_\ell,B_1,\ldots,B_\ell) \in \mathcal{R}(k)$, the setting

$$\xi_R = \begin{cases} \dfrac{1}{k+1} & \text{if } \ell\in\{1,k\}\\ 0 & \text{otherwise}\end{cases}$$

is a such feasible primal solution, with objective function value $\lambda = \frac{2}{k+1}$. In fact, this is the primal solution corresponding to the family of feasible sets used in the proof of Theorem 2. So Theorem 4 will follow from Lemma 3 below. However, in order to establish the claim of Theorem 5, we need to show that there exists a primal solution with $\xi_R = 0$ for all $\ell > 2$. This is essentially the content of Lemma 2 below.

Proof of Theorems 4 and 5. For positive integers $u, v \in \mathbb{Z}_+$ denote by $p(u,v)$ the number of ways to partition u into v non-negative numbers, i.e. the number of *unordered* v-tuples $(r_1,\ldots,r_v) \in \mathbb{Z}_+^v$ such that $\sum_{i=1}^v r_i = u$. For each such tuple $(r_1,\ldots,r_v)$, let $\chi(r_1,\ldots,r_v)$ denote the number of all possible distinct *ordered* sequences of length v, formed by considering each r_i as a symbol. For example, if $u = 4$ and $v = 3$, we have $p(u,v) = 4$ partitions, namely $(4,0,0)$, $(3,1,0)$, $(2,2,0)$ and $(2,1,1)$. The corresponding numbers of sequences are $\chi(4,0,0) = 3$, $\chi(3,1,0) = 6$, $\chi(2,2,0) = 3$ and $\chi(2,1,1) = 3$. Note that, for a pair of integers (r_1,r_2), $\chi(r_1,r_2) = 1$ if $r_1 = r_2$ and $\chi(r_1,r_2) = 2$ otherwise. Also, for $k \in \mathbb{Z}_+$, we have $p(k,2) = \lfloor k/2\rfloor + 1$. Thus

$$\sum_{r=0}^{\lfloor\frac{k}{2}\rfloor-1} \chi(r,k-r-2) = k-1. \tag{3}$$

By $N(\ell,r_1,\ldots,r_\ell)$ we denote the number of elements $R = (p_1,\ldots,p_\ell,B_1,\ldots,B_\ell)$ of $\mathcal{R}(k)$ such that $|B_i| = r_i$ for $i = 1,\ldots,\ell$. Thus

$$N(\ell,r_1,\ldots,r_\ell) = \binom{k}{\ell}\binom{k-\ell}{r_1,\ldots,r_\ell} = \binom{k}{\ell,r_1,\ldots,r_\ell}.$$

Theorems 4 and 5 follow from the next Lemmas 2 and 3 below.

Lemma 2 *For $R = (p_1,\ldots,p_\ell,B_1,\ldots,B_\ell\} \in \mathcal{R}(k)$, the setting*

$$\xi_R = \begin{cases} \dfrac{2}{k(k+1)} & \text{if } \ell = 1 \\ \dfrac{1}{(k+1)N(2,|B_1|,|B_2|)} & \text{if } \ell = 2 \\ 0 & \text{if } \ell > 2 \end{cases}$$

is a feasible solution to the dual LP (2), with objective function value $\lambda = \frac{2}{k+1}$.
Proof: First we verify that $\sum_{R\in\mathcal{R}(k)} \xi_R = 1$. For this we note that

$$\begin{aligned}\sum_{R\in\mathcal{R}(k)} \xi_R &= \sum_{(p_1,B_1)\in\mathcal{R}(k)} \frac{2}{k(k+1)} + \sum_{(p_1,p_2,B_1,B_2)\in\mathcal{R}(k)} \frac{1}{(k+1)N(2,|B_1|,|B_2|)} \\ &= \frac{2N(1,k-1)}{k(k+1)} + \sum_{r=0}^{\lfloor \frac{k}{2} \rfloor -1} \frac{|\{(p_1,p_2,B_1,B_2)\in\mathcal{R}(k):\ |B_1|=r \text{ or } |B_2|=r\}|}{(k+1)N(2,r,k-r-2)} \\ &= \frac{1}{k+1}\left(2 + \sum_{r=0}^{\lfloor \frac{k}{2} \rfloor -1} \chi(r,k-r-2)\right) = 1,\end{aligned}$$

where the last equality follows from (3). Now it remains to show that

$$\sum_{R\in\mathcal{R}(k)} a_{R,C}\xi_R \geq \lambda = \frac{2}{k+1}, \text{ for all } C\in\mathcal{C}(k). \tag{4}$$

For an element $R = (p_1,\ldots,p_\ell,B_1,\ldots,B_\ell) \in \mathcal{R}(k)$ with $|B_i| = r_i$ for $i = 1,\ldots,\ell$, it is easy to see that the number of subsets C of size $|C| = i$ for which $a_{R,C} = 1$ is

$$N'(i,\ell,r_1,\ldots,r_\ell) = \sum_{j=1}^{\ell} \binom{r_j}{i-1}.$$

Fix integers $i,\ell,r_1\ldots,r_\ell$. Let $\mathcal{R}' = \mathcal{R}'(\ell,r_1,\ldots,r_\ell) \subseteq \mathcal{R}(k)$ be a subset of the rows of $A(k)$, such that for each $R\in\mathcal{R}'$ the number of parents in R is ℓ and the distribution of the remaining $k-\ell$ elements among these parents is $r_1,\ldots,r_\ell$ (i.e. if $R = (p_1,\ldots,p_\ell,B_1,\ldots,B_\ell) \in \mathcal{R}'$, then the sequence $(|B_1|,\ldots,|B_\ell|)$ is a permutation of $(r_1,\ldots,r_\ell)$). Clearly, not every subset $C\in\mathcal{C}(k)$ of size $|C| = i$ is covered the same number of times by a certain row $R\in\mathcal{R}'$, i.e. there exist subsets $C,C'\in\mathcal{C}(k)$ such that $|C| = |C'| = i$ and yet $a_{R,C} \neq a_{R,C'}$. However, if we consider all rows in $\mathcal{R}'$, then it follows by symmetry that every subset $C\in\mathcal{C}(k)$ of size $|C| = i$ is covered the same number of times by these rows. In other words, there is a single number $N'' = N''(i,\ell,r_1,\ldots,r_\ell)$ such that $\sum_{R\in\mathcal{R}'} a_{R,C} = N''$ for all $C\in\mathcal{C}(k)$ such that $|C| = i$. Since the total number of elements of $\mathcal{C}(k)$ of size i is $\binom{k}{i}$ it follows that

$$\begin{aligned}\binom{k}{i}N''(i,\ell,r_1,\ldots,r_\ell) &= \sum_{C\in\mathcal{C}(k):|C'|=i}\ \sum_{R\in\mathcal{R}'} a_{R,C'} = \sum_{R\in\mathcal{R}'}\ \sum_{C\in\mathcal{C}(k):|C'|=i} a_{R,C'} \\ &= |\mathcal{R}'|N'(i,\ell,r_1,\ldots,r_\ell) \\ &= \chi(r_1,\ldots,r_\ell)N(\ell,r_1,\ldots,r_\ell)N'(i,\ell,r_1,\ldots,r_\ell),\end{aligned}$$

from which

$$N''(i,\ell,r_1,\ldots,r_\ell) = \frac{\chi(r_1,\ldots,r_\ell)N(\ell,r_1,\ldots,r_\ell)N'(i,\ell,r_1,\ldots,r_\ell)}{\binom{k}{i}} \tag{5}$$

follows. Now to show (4), fix a $C \in \mathcal{C}(k)$ of size $|C| = i$. Then

$$\sum_{R\in\mathcal{R}(k)} a_{R,C}\xi_R = \sum_{R=(p_1,B_1)\in\mathcal{R}(k)} \frac{2a_{R,C}}{k(k+1)} + \sum_{R=(p_1,p_2,B_1,B_2)\in\mathcal{R}(k)} \frac{a_{R,C}}{(k+1)N(2,|B_1|,|B_2|)}. \tag{6}$$

The first part of right-hand side of (6) is equal to

$$S_1 = \frac{2}{k(k+1)}N''(i,1,k-1) = \frac{2\binom{k-1}{i-1}}{(k+1)\binom{k}{i}}. \tag{7}$$

The second part is equal to

$$\begin{aligned}
S_2 &= \sum_{r=0}^{\lfloor\frac{k}{2}\rfloor-1} \sum_{R\in\mathcal{R}'(2,r,k-r-2)} \frac{a_{R,C}}{(k+1)N(2,r,k-r-2)} \\
&= \sum_{r=0}^{\lfloor\frac{k}{2}\rfloor-1} \frac{1}{(k+1)N(2,r,k-r-2)} \sum_{R\in\mathcal{R}'(2,r,k-r-2)} a_{R,C} \\
&= \sum_{r=0}^{\lfloor\frac{k}{2}\rfloor-1} \frac{N''(i,2,r,k-r-2)}{(k+1)N(2,r,k-r-2)} \\
&= \sum_{r=0}^{\lfloor\frac{k}{2}\rfloor-1} \frac{\chi(r,k-r-2)N'(i,2,r,k-r-2)}{(k+1)\binom{k}{i}} \\
&= \sum_{r=0}^{\lfloor\frac{k}{2}\rfloor-1} \chi(r,k-r-2)\frac{\binom{r}{i-1}+\binom{k-r-2}{i-1}}{(k+1)\binom{k}{i}} \\
&= \frac{2\sum_{r=0}^{k-2}\binom{r}{i-1}}{(k+1)\binom{k}{i}}.
\end{aligned} \tag{8}$$

Now summing (7) and (8) gives

$$\sum_{R\in\mathcal{R}(k)} a_{R,C}\xi_R = S_1 + S_2 = \frac{2}{k+1}\cdot\frac{\sum_{r=0}^{k-1}\binom{r}{i-1}}{\binom{k}{i}} = \frac{2}{k+1},$$

using a well-known combinatorial identity [8]. This shows (4). □

Lemma 3 *For $C \in \mathcal{C}(k)$, the setting*

$$z_C = \begin{cases} \dfrac{1}{\binom{k+1}{2}} & \text{if } |C| \le 2 \\ 0 & \text{if } |C| > 2 \end{cases}$$

is a feasible solution to the dual LP (2), with objective function value $\lambda = \frac{2}{k+1}$.

Proof: Clearly, $z \geq 0$ and

$$\sum_{C \in \mathcal{C}(k)} z_C = \frac{1}{\binom{k+1}{2}} \left[\binom{k}{1} + \binom{k}{2} \right] = 1,$$

so all what we need to check is that

$$\sum_{C \in \mathcal{C}(k)} a_{R,C} z_C \leq \lambda = \frac{2}{k+1}, \text{ for all } R \in \mathcal{R}(k). \tag{9}$$

Again we show that (9) holds with equality. Fix $R = (p_1, \ldots, p_\ell, B_1, \ldots, B_\ell) \in \mathcal{R}(k)$ with $|B_i| = r_i$ for $i = 1, \ldots, \ell$. Then

$$\begin{aligned}
\sum_{C \in \mathcal{C}(k)} a_{R,C} z_C &= \frac{1}{\binom{k+1}{2}} \sum_{C \in \mathcal{C}(k):|C|=1} a_{R,C} + \frac{1}{\binom{k+1}{2}} \sum_{C \in \mathcal{C}(k):|C|=2} a_{R,C} \\
&= \frac{1}{\binom{k+1}{2}} N'(1, \ell, r_1, \ldots, r_\ell) + \frac{1}{\binom{k+1}{2}} N'(2, \ell, r_1, \ldots, r_\ell) \\
&= \frac{2}{k(k+1)} \ell + \frac{2}{k(k+1)} \sum_{j=1}^{\ell} \binom{r_j}{1} = \frac{2\ell}{k(k+1)} + \frac{2(k-\ell)}{k(k+1)} = \frac{2}{k+1}.
\end{aligned}$$

□

3.4. *APX-hardness for $k \geq 2$*

Recall that completeness within APX is defined through L reductions, see for instance [18]. So an approximation scheme for an APX-complete problem translates into such a scheme for any problem in APX.

Theorem 6 *For each $k \geq 2$,* SIM-MATCH *with k constraint sets is APX-hard.*

Proof: We only have to modify our reduction from the proof of Theorem 1 slightly to account for the new setting. Instead of testing for a given number ℓ of variables x_i, we let $\ell = p$, the cardinality of $\mathcal{C}$. So a perfect solution would have to pack all sets into U. In order to get an approximation-preserving reduction, we need to make sure that a certain fraction of the sets can always be packed. This is achieved by restricting to 3-SET-PACKING(2), which is already APX-hard [2].

In this situation, the overall number of variables is at most $9p$ since there are p choice variables, and each gadget contributes at most 8 variables. Let M denote the number of gadget variables; then $M \leq 8p$. Since each element of U appears in at most 2 sets and since each set is of size at most 3, we can always find at least $p/4$ disjoint sets. (Just construct any maximal collection of disjoint sets. Including any one set in the collection rules out inclusion of at most 3 other sets.) Thus, if the optimal set packing has k_{opt} sets then $k_{\text{opt}} \geq p/4$.

Let s_{opt} denote the value of an optimal solution to the SIM-MATCH instance constructed. Note that s_{opt} counts variables, while k_{opt} counts sets. We claim that $s_{\text{opt}} = k_{\text{opt}} + M$. The relation "$\geq$" follows simply from assigning the k_{opt} input values of the gadgets that correspond to an optimal packing to some x_i. Then all gadget variables can be assigned values without conflict. To see "$\leq$," notice that any assignment can be transformed into one in which all gadget variables receive values, without decreasing the total number of

satisfied variables. It is then easy to see that we can find a set packing with as many sets as we have x_i assigned with values. This shows the claim.

Suppose now that SIM-MATCH can be approximated within a factor of α. That is, we can find in polynomial time a feasible assignment on s variables, where s is at least αs_{opt}. Then $s = k' + M \geq \alpha(k_{opt} + M)$, so $k' \geq \alpha k_{opt} - M(1-\alpha) \geq \alpha k_{opt} - 8p(1-\alpha) \geq \alpha k_{opt} - 8(4k_{opt})(1-\alpha) = k_{opt}(33\alpha - 32)$. Thus, an α-approximation for SIM-MATCH gives a $(33\alpha - 32)$-approximation for 3-SET-PACKING(2). This gives the desired L reduction. □

Plugging in the current-best known inapproximability bound of 99/100 for 3-SET-PACKING(2) from [3] into the above reduction, we learn that SIM-MATCH cannot be approximated to within a factor of $1 - 1/3300$ unless P = NP.

4. The SIMULTANEOUS MATCHINGS polytope

Consider again instances of SIM-P-MATCH, on complete bipartite graphs, with $k = 2$. As remarked in Section 2, checking feasibility in such a setting is trivial. In [1], a somewhat different aspect of this setting is considered. Assume that the set D is labelled by the set of integers $0, 1, \ldots, d-1$, and $X = \{x_1, x_2, \ldots, x_n\}$. Then every feasible solution becomes an integer vector in the n-dimensional space $\{0, 1, \ldots, d-1\}^n$. Now what is the structure of the polytope defined by the convex hull of integer vectors corresponding to feasible solutions? The authors of [1] establish the dimension of this polytope and also obtain classes of facet-defining inequalities.

We consider the variant where dimensions / variables are associated with each edge of the graph, rather than each vertex in X. Viewed as a purely graph-theoretic decision / optimisation problem, this makes eminent sense as it directly generalises the well-studied matching polytope (see for instance [12]): we wish to assign 0,1 values to each edge variable (a value of 1 for an edge corresponds to putting this edge into the solution M, 0 corresponds to omitting this edge) such that all vertices of X (or as many as possible) have an incident edge in M, and M is a feasible solution. This is easy to write as an integer program:
Choose x_e for each edge e so as to

$$\begin{aligned} \text{maximise} \quad & \sum_{e \in E} w_e x_e \quad \text{(for SIM-W-MATCH)} \\ S.T. \quad & \forall x \in X : \sum_{e=(x,z): z \in D} x_e \leq 1, \qquad \forall z \in D : \sum_{e=(x,z): x \in X_1} x_e \leq 1, \\ & \forall z \in D : \sum_{e=(x,z): x \in X_2} x_e \leq 1, \qquad \forall e : x_e \in \{0,1\} \end{aligned}$$

The corresponding linear program replaces the last condition above by $\forall e : x_e \in [0,1]$. Let P_I denote the convex hull of integer solutions to the integer program, and let P_L denote the convex hull of feasible solutions to the linear program. P_I and P_L are polytopes in $\mathbb{R}^n$, with $P_I \subseteq P_L$.

The special case of the above where there is just one constraint set (either X_1 or X_2 is empty) is the bipartite matching polytope. For this polytope, it is known that every vertex is integral; i.e. $P_I = P_L$. For non-bipartite graphs, this polytope is not necessarily integral, but it is known that all vertices there are half-integral (i.e. at any extremal point

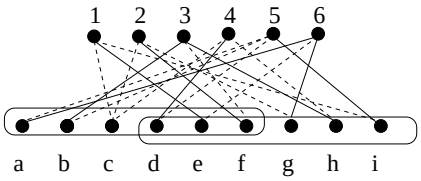

Fig. 4. A vertex of P_L that is not half-integral

of the polytope, all edge weights are from the set $\{0, 1/2, 1\}$). Unfortunately, these nice properties break down even for two constraint sets. We illustrate this with an example in Figure 4. The underlying graph is the complete bipartite graph. Assign weights of 1/3 to the edges shown by dotted lines, 2/3 to those shown with solid lines, and 0 to all other edges. This gives a feasible solution and hence a point in P_L, and it can be verified [4] that it is in fact a vertex of P_L and is outside P_I.

Acknowledgment

We thank Mahmoud Fouz and David Steurer for pointing out the simple feasible family used in the proof of Theorem 2 and for helpful discussions.

References

[1] G. Appa, D. Magos, and I. Mourtos. On the system of two all_different predicates. *Information Processing Letters*, 94/3:99–105, 2005.
[2] P. Berman and T. Fujito. Approximating independent sets in degree 3 graphs. In *WADS 1995*, volume 955 of *LNCS*, pages 449–460, 1995.
[3] M. Chlebík and J. Chlebíková. Inapproximability results for bounded variants of optimization problems. In *FCT 2003*, volume 2751 of *LNCS*, pages 27–38, 2003.
[4] J. Edmonds. Path, trees and flowers. *Canadian Journal of Mathematics*, pages 233–240, 1965.
[5] M. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. Freeman, 1979.
[6] D. S. Hochbaum, editor. *Approximation Algorithms for NP-Hard Problems*. Brooks/Cole Pub. Co., 1996.
[7] M. Karpinski and W. Rytter. *Fast parallel algorithms for graph matching problems*. 1998. Oxford Lecture Series in Mathematics and its Applications 9.
[8] D. E. Knuth. *Fundamental Algorithms*, volume 1 of *The Art of Computer Programming*, section 1.2, pages 10–119. Addison-Wesley, Reading, Massachusetts, second edition, 10 January 1973.
[9] H.W. Kuhn. The Hungarian Method for the assignment problem. *Naval Research Logistic Quarterly*, 2:83–97, 1955.
[10] M. Leconte. A bounds-based reduction scheme for constraints of difference. In *Proceedings of the Constraint-96 Workshop*, pages 19–28, 1996.
[11] A. Lopez-Ortiz, C.-G. Quimper, J. Tromp, and P. van Beek. A fast and simple algorithm for bounds consistency of the alldifferent constraint. In *IJCAI*, 2003.
[12] L. Lovasz and M. Plummer. *Matching Theory*. North-Holland, 1986. Annals of Discrete Mathematics 29.
[13] K. Mehlhorn and S. Thiel. Faster Algorithms for Bound-Consistency of the Sortedness and the AllDifferent Constraint. In *CP*, 2000.
[14] PORTA. http://www.zib.de/optimization/software/porta/.

[4] The software PORTA [14] was used to find this vertex.

[15] J.-F. Puget. A fast algorithm for the bound consistency of alldiff constraints. In *AAAI*, 1998.
[16] Jean-Charles Régin. A filtering algorithm for constraints of difference in CSPs. In *AAAI*, pages 362–367, 1994.
[17] W.J. van Hoeve. The AllDifferent Constraint: A Survey. In *Proceedings of the Sixth Annual Workshop of the ERCIM Working Group on Constraints*, 2001.
[18] V. Vazirani. *Approximation Algorithms*. Springer, 2001.