Simultaneous Matchings

Khaled Elbassioni[1], Irit Katriel[2], Martin Kutz[1], and Meena Mahajan[3]

[1] Max-Plank-Institut für Informatik, Saarbrücken, Germany.
{elbassio,mkutz}@mpi-sb.mpg.de
[2] BRICS University of Aarhus, Åbogade 34, Århus, Denmark. irit@daimi.au.dk
[3] The Institute of Mathematical Sciences, Chennai, India. meena@imsc.res.in

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 saturates 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 $O(2^k \mathrm{poly}(k, |X \cup D|))$ time.

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, 11].

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-cardinality matchings, we may relax the perfect matching condition and ask for a largest 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 (see, e.g., [9, 10, 12, 14, 15]). 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 $V = X \cup D$ and $E \subseteq X \times D$, a weight or profit $w(e)$ associated with each edge in E, and a collection of constraint sets $\mathcal{F} \subseteq 2^X$.

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[4]; 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 [8] 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

[4] 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.

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.
k-SET-PACKING: The restriction where every set in $\mathcal{C}$ contains at most k elements.
SET-PACKING(r): The restriction where every vertex 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 to variables (as in the constraint programming application described in Section 1). Each element a from the universe U is embodied by a single value v_a and there are ℓ variables $x_1, \ldots, x_\ell$, which belong to both constraint sets X_1 and X_2, that are fighting for these values. Between the x_i's and the v_j's we place gadgets that encode the sets in $\mathcal{C}$.

Consider the trivial situation with only singleton sets in $\mathcal{C}$. There we could simply connect each x_i 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.

The difficult part is to build gadgets between the x_i and U such that a single variable occupies more than one value from U. Therefore consider 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 color indicating membership in both sets). If the leftmost value is assigned to some red-green

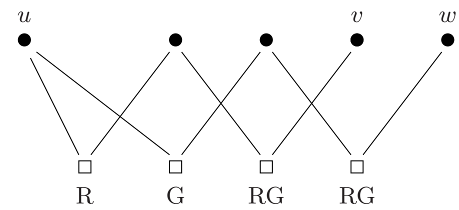

Fig. 1. The basic gadget for the reduction.

variable outside the figure then the two left variables will be forced to claim the two values to their right and 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 merge the 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.)

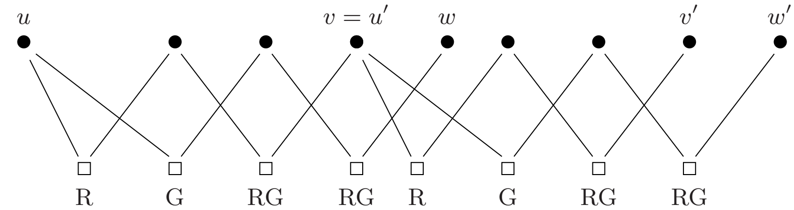

Fig. 2. Merging two 1:2-gadgets into one 1:3-gadget.

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. Then we connect each red-green variable x_i to the input value of each such set gadget. The resulting configuration has obviously the desired behaviour. We can assign values to all variables without violating the red and green constraint if and only if we can pack ℓ sets from $\mathcal{C}$ into U. See the appendix for a complete example. $\square$

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 the vertices of $X_2 \setminus X_1$ with vertices that were not matched with vertices from the intersection $X_1 \cap X_2$. 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 the following NP-complete problem, known as 3-VERTEX-COLORING (see for instance[5]):

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

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 $\mathcal{F} = \{(u, v) \mid (u, v) \in A\}$. It is straightforward to see that any feasible solution to this instance of SIM-P-MATCH is a 3-coloring 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 $\epsilon > 0$ and a polynomial-time algorithm A such that for every instance x of Π we have $(1 - \epsilon) \cdot \mathrm{Opt}(\Pi, x) \leq A(x) \leq \mathrm{Opt}(\Pi, x)$. Clearly, the smaller the ϵ, the better the quality of approximation. The factor $1 - \epsilon$ is referred to as the approximation factor achieved by algorithm A. For more details, see any text on approximation algorithms, such as [6, 16].

Consider the following naive polynomial-time approximation algorithm for SIM-W-MATCH: Find a maximum profit matching for each constraint set $C \in \mathcal{F}$ independently and return the most profitable 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 constraints is in APX.*

3.2 Improving the approximation factor

We can slightly improve the $1/k$ factor by considering more than one set X_i at a time. Fix a maximum-weight feasible solution M with profit Opt. For any set $S \subseteq X$, let $f(S)$ denote the profit of the maximum weight simultaneous matching in the graph induced by $S \cup D$, and let $g(S)$ be the profit 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 the case with two constraint sets X_1, X_2. We can compute each of $f(X_1)$ and $f(X_2)$, 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. Alltogether 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 generalise the notion of symmetric differences appropriately. Define $\hat{X}_i := X_i \setminus \bigcup_{h\neq i} X_h$ and consider the reduced pairs $S_{ij} = \hat{X}_i \cup \hat{Y}_i$. As before, we can determine $f(S_{ij})$ exactly for any index pair $i \neq j$ by independent computation of maximum matchings on $\hat{X}_i$ and $\hat{X}_j$. A careful choice of coefficients yields

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

and again by averaging, at least one of the $f(X_i)$ or $f(S_{ij})$ is no less than $4/(3k) \cdot \mathrm{Opt}$.

Proposition 3. *For any constant k,* SIM-W-MATCH *with k constraint sets can be approximated in polynomial time within a factor of $4/(3k)$.*

We now pursue this approach to its logical conclusion. The main idea is to identify subsets S of X for which (a) $f(S)$ is upper bounded by Opt and (b) $f(S)$ can be efficiently computed. Note that for any $S \subseteq X$, (a) comes for free, since in computing $f(S)$ we consider all the original constraint sets, restricted to S. Then, for every choice of weights ξ_S, we have

$$\sum_S \xi_S f(S) \geq \sum_S \xi_S g(S) \geq F_\xi \cdot \mathrm{Opt}$$

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 proving Proposition 3, the chosen S's were the X_i's and the S_{ij}, and $F_\xi = 2$ and $T_\xi = 3k/2$.)

Clearly, we lose nothing by considering only *maximal* subsets S for which $f(S)$ is efficiently computable. (The sets S_{ij} above were not maximal in this sense.) Below, we consider only maximal subsets. Our approach can be sketched as follows.

Let $\mathcal{R}$ denote the collection of maximal sets $S \subseteq X$ for which $f(S)$ can be efficiently computed (and is upper bounded by Opt). Also, let $\mathcal{C}$ denote the family of maximal subsets of X entirely contained in either of each X_i or $X \setminus X_i$. 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, $\beta = |\mathcal{C}| = 2^k - 1$. We characterise the sets in $\mathcal{R}$ and show that $\alpha = |\mathcal{R}|$ is $O((2k)^k)$. Also, we note that for each $R \in \mathcal{R}$ and each $C \in \mathcal{C}$, C is either completely contained in or completely outside R. We now wish to compute, for each $R \in \mathcal{R}$, a weight ξ_R such that the corresponding ratio F_ξ / T_ξ is maximised. Define an $\alpha \times \beta$ 0-1 matrix A where $a_{R,C} = 1$ iff $C \subseteq R$. Consider the following pair of primal-dual linear programs:

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

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 2. *For any integer $k \geq 1$, we have $\lambda^*_P(k) = \lambda^*_D(k) = \frac{2}{k+1}$. There is a primal optimal solution $\xi \in \mathbb{R}^\alpha$ with support $|\{i \in [\alpha] \ : \ \xi_i > 0\}| = k + \binom{k}{2} 2^{k-2}$.*

Due to space limitations, we skip the proof of this theorem; it can be found in the appendix / the full version. Theorem 2 establishes that an approximation factor of $2/(k+1)$ is possible. To compute the running time of the approximation, note that each $f(R)$ can be computed in polynomial time, and we need to compute $f(R)$ only for those $R \in \mathcal{R}$ having non-zero ξ_R. Hence by Theorem 2, the running time is polynomial provided $k + \binom{k}{2} 2^{k-2}$ is polynomial in the input size.

Theorem 3. SIM-W-MATCH *can be approximated within a factor of $2/(k+1)$ by an algorithm that runs in $O(2^k \mathrm{poly}(n, m, d, k))$ time.*

Thus, for all instances of SIM-W-MATCH with $k = O(\log N)$, where $N = \max\{n, m, d\}$, a $\frac{2}{k+1}$ approximation can be found in polynomial time. Furthermore, Theorem 2 tells us also that this factor is the best-possible via the above approach.

3.3 APX-hardness for $k \geq 2$

An optimisation problem Π is APX-hard if for every problem Π' in APX there is a reduction from Π' to Π preserving constant-factor approximability. The reduction must be such that any α-factor approximation for Π can be converted into a β-factor approximation for Π', where β may depend only on α and not on the input or only even the input size; and for every given approximation quality $\beta < 1$ there is an $\alpha < 1$ such that an α-approximation of for Π gives a β-approximation for Π'. We show below that SIM-MATCH is also APX-hard.

Theorem 4. *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 have 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 $1-\epsilon = \alpha$. 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). ☐

For every APX-complete maximisation problem Π, there is a constant $\gamma > 0$ such that if Π can be approximated to a constant factor strictly larger than γ, then P=NP. (This follows from the PCP theorem. See for instance [6, 16].) This γ is referred to as the inapproximability threshold for Π. Let β be the inapproximability threshold for 3-SET-PACKING(2). The reduction described above converts any α-factor approximation algorithm for SIM-MATCH to a $(33\alpha - 32)$ factor approximation algorithm for 3-SET-PACKING(2). Thus, unless P=NP, it must be that $(33\alpha - 32) \leq \beta$ and so $\alpha \leq (\beta + 32)/33$. Thus the inapproximablity threshold for SIM-MATCH is at least $(\beta + 32)/33$. The currently best known lower bound for β, the inapproximability threshold for 3-SET-PACKING(2), is 100/99 [3]. So SIM-MATCH cannot be approximated to a constant better than 3268/3267.

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 $X = \{x_1, x_2, \ldots, x_n\}$. Each feasible solution is now an integer vector in n-dimensional space, in $\{0, 1, \ldots, d-1\}^n$. 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 [11]): 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

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

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 $P_I \subseteq P_L$.

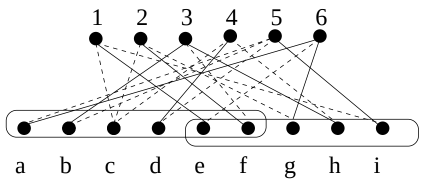

Fig. 3. A vertex of P_L that is not half-integral

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 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 3. 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 dashed lines, and 0 to all other edges. This gives a feasible solution and hence a point in P_L, and it can be verified[5] that it is in fact a vertex of P_L and is outside P_I.

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. Dorit S. Hochbaum, editor. *Approximation Algorithms for NP-Hard Problems.* Brooks/Cole Pub. Co., 1996.
7. Marek Karpinski and Wojciech Rytter. *Fast parallel algorithms for graph matching problems.* Oxford University Press, Oxford, 1998. Oxford Lecture Series in Mathematics and its Applications 9.
8. H.W. Kuhn. The Hungarian Method for the assignment problem. *Naval Research Logistic Quarterly*, 2:83–97, 1955.
9. M. Leconte. A bounds-based reduction scheme for constraints of difference. In *Proceedings of the Constraint-96 International Workshop on Constraint-Based Reasoning*, pages 19–28, 1996.
10. 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 *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, 2003.
11. L Lovasz and M Plummer. *Matching Theory.* North-Holland, 1986. Annals of Discrete Mathematics 29.
12. K. Mehlhorn and S. Thiel. Faster Algorithms for Bound-Consistency of the Sortedness and the Alldifferent Constraint. In *CP 2000*, volume 1894 of *LNCS*, pages 306–319, 2000.
13. PORTA. http://www.zib.de/optimization/software/porta/.
14. J.-F. Puget. A fast algorithm for the bound consistency of alldiff constraints. In *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, pages 359–366, July 1998.
15. W.J. van Hoeve. The Alldifferent Constraint: A Survey. In *Proceedings of the Sixth Annual Workshop of the ERCIM Working Group on Constraints*, 2001. `http://www.arxiv.org/html/cs/0110012`.
16. Vijay Vazirani. *Approximation Algorithms.* Springer, 2001.

[5] The software PORTA [13] was used to find this vertex.

Appendix

A complete example for the NP-hardness reduction

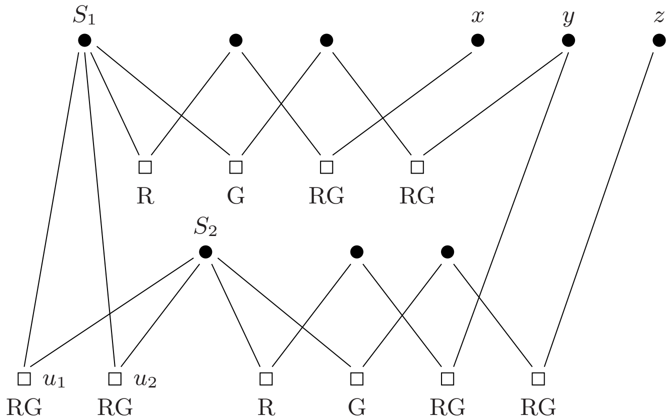

Fig. 4. The SIM-P-MATCH instance obtained from the unsolvable 3-SET-PACKING(2) instance $S_1 = \{x, y\}$, $S_2 = \{y, z\}$ and $\ell = 2$.

Proof of Theorems 2 and 3

Let $k \geq 1$ be a given positive integer, and denote by $[k]$ the set $\{1, \ldots, k\}$. Let $\mathcal{R}(k)$ be the set of all possible sequences obtained from the elements of the set $[k]$ in the following way: (i) Select a subset $P \subseteq [k]$ of size ℓ, where $\ell \in [k]$. The elements of P are called the *parent* elements. 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 $[k]$ exactly one parent from 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). Let $\mathcal{C}(k) = 2^{[k]} \setminus \{\emptyset\}$ 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 in the partition defined by R, i.e. $C \subseteq \{p_j\} \cup B_j$ for some $j \in [\ell]$. Write $\alpha = |\mathcal{R}|$, $\beta = |\mathcal{C}|$ and $A = A(k)$, and consider the pair of primal-dual linear programs given by (1) over $\xi \in \mathbb{R}^\alpha$ and $y \in \mathbb{R}^\beta$.

Proof of Theorem 3. We show how Theorem 2 can be used to obtain a factor $2/(k+1)$-approximation algorithm for SIM-W-MATCH. Given an instance $(X, D, E, \mathcal{F}, w)$ of SIM-W-MATCH, we construct for each $R \in \mathcal{R}(k)$ a corresponding family $\mathcal{F}(R)$ as follows. Assume 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' that is another parent or is assigned to another parent. 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 $\mathcal{F}(R) = \{X_{p_i}(R) \ : \ i = 1, \ldots, \ell\}$. 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 *non-conflicting* (i.e. there is no set $X_i \in \mathcal{F}$ intersecting two distinct sets in $\mathcal{F}(R)$): 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 size 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. It is not difficult to see that the families $\{\mathcal{F}(R) \ : \ R \in \mathcal{R}(k)\}$ are exactly those maximal non-conflicting families, i.e. those satisfying conditions (a) and (b) stated in Section 3.2.

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 $g(R) \leq f(R) \leq \mathrm{Opt}$. 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. Theorem 3 follows from the following lemma.

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

Proof. Let $\xi_1, \ldots, \xi_\alpha$ be an optimal solution to the primal linear program (1).

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

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

On the other hand,

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

a contradiction. □

Proof of Theorem 2. 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 $(r_1, \ldots, r_v) \in \mathbb{Z}_+$ 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 $(2,2,1)$. The corresponding numbers of sequences are $\chi(4,0,0) = 3$, $\chi(3,1,0) = 6$, $\chi(2,2,0) = 3$ and $\chi(2,2,1) = 3$. 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. Thus

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

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

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

Theorem 2 follows from the next two lemmas.

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

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

is a feasible solution to the dual LP (1), 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 (2). Now it remains to show that

$$\sum_{R \in \mathcal{R}(k)} a_{R,C} \xi_R \geq \lambda = \frac{2}{k+1}, \text{ for all } C \in \mathcal{C}(k). \tag{3}$$

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 $R \in \mathcal{R}(k)$, 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

$$N''(i,\ell,r_1,\ldots,r_\ell) = \frac{\chi(r_1,\ldots,r_\ell)N(\ell,r_1,\ldots,r_\ell)N'(i,\ell,r_1,\ldots,r_\ell)}{\binom{k}{i}} \tag{4}$$

follows. Now to show (3), fix a $C \in \mathcal{C}(k)$ of size $|C| = i$. Then

$$\sum_{R\in\mathcal{R}(k)} a_{R,C}\xi_R = \sum_{R=(p_1,B_1)\in\mathcal{R}(k)} \frac{2a_{R,C}}{k(k+1)} + \sum_{R=(p_1,p_2,B_1,B_2)\in\mathcal{R}(k)} \frac{a_{R,C}}{(k+1)N(2,|B_1|,|B_2|)}. \tag{5}$$

The first part of right-hand side of (5) is equal to

$$S_1 = \frac{2}{k(k+1)}N''(i,1,k-1) = \frac{2\binom{k-1}{i-1}}{(k+1)\binom{k}{i}}. \tag{6}$$

The second part is equal to

$$\begin{aligned}S_2 &= \sum_{r=0}^{\lfloor\frac{k}{2}\rfloor-1}\sum_{R\in\mathcal{R}'(2,r,k-r-2)} \frac{a_{R,C}}{(k+1)N(2,r,k-r-2)}\\ &= \sum_{r=0}^{\lfloor\frac{k}{2}\rfloor-1} \frac{1}{(k+1)N(2,r,k-r-2)}\sum_{R\in\mathcal{R}'(2,r,k-r-2)} a_{R,C}\\ &= \sum_{r=0}^{\lfloor\frac{k}{2}\rfloor-1} \frac{N''(i,2,r,k-r-2)}{(k+1)N(2,r,k-r-2)}\\ &= \sum_{r=0}^{\lfloor\frac{k}{2}\rfloor-1} \frac{\chi(r,k-r-2)N'(i,2,r,k-r-2)}{(k+1)\binom{k}{i}}\\ &= \sum_{r=0}^{\lfloor\frac{k}{2}\rfloor-1} \chi(r,k-r-2)\frac{\binom{r}{i-1}+\binom{k-r-2}{i-1}}{(k+1)\binom{k}{i}}\\ &= \frac{2\sum_{r=0}^{k-2}\binom{r}{i-1}}{(k+1)\binom{k}{i}}.\end{aligned} \tag{7}$$

Now summing (6) and (7) 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. This shows (3). □

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

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

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

Proof. Clearly, $y \ge 0$ 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

$$\sum_{C \in \mathcal{C}(k)} a_{R,C} z_C \le \lambda = \frac{2}{k+1}, \text{ for all } R \in \mathcal{R}(k). \tag{8}$$

Again we show that (8) 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

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

□