Simultaneous Matchings

Khaled Elbassioni\textsuperscript{1}, Irit Katriel\textsuperscript{2}, Martin Kutz\textsuperscript{1}, and Meena Mahajan\textsuperscript{**3}

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

\textbf{Abstract} \ Given a bipartite graph $G = (X \cup D, E \subseteq X \times D)$, an \textit{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 $2^{\text{poly}(k,|X \cup D|)}$ time.

\section{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 $V = X \cup D$ (so $E \subseteq X \times D$) and a collection $\mathcal{F} \subseteq 2^X$ of $k$ \textit{constraint sets}. A solution to the problem is a subset $M \subseteq E$ of the edges such that $M$ is \textit{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

\textsuperscript{*} Supported by the Danish Research Agency (grant # 272-05-0081).
\textsuperscript{**} This work was done while the fourth author was visiting the Max-Plank-Institut für Informatik, Saarbrücken, Germany.
\textsuperscript{***} Basic Research in Computer Science, funded by the Danish National Research Foundation.
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 \textit{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 \textit{constraints} that specify which combinations of assignments of values to variables are permitted.

An extensively studied constraint is the \textit{AllDifferent} constraint (\textit{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 \textit{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 \textit{AllDiff} constraints, defined over possibly overlapping variable sets. This setting corresponds to the generalisation we propose.

Formally, we consider the following problems:

\textbf{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.

\textbf{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 \textit{SIM-MATCH} and \textit{SIM-P-MATCH} respectively. In this case, the decision version of \textit{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

\(^4\) 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 \textit{SIM-W-PERF-MATCH} is not a special case of \textit{SIM-W-MATCH}.
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
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 $k$ con-
straint 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 $C = \{S_1, S_2, \ldots, S_p\}$ of subsets of $U$.

Decision problem: Given an integer $\ell \leq p$, is there a collection $C' \subseteq C$ of at least
$\ell$ pairwise disjoint sets?

Optimisation problem: Find a collection $C' \subseteq C$ of pairwise disjoint subsets such
that $|C'|$, the number of chosen subsets, is maximised.

$k$-SET-PACKING: The restriction where every set in $C$ contains at most $k$ ele-
ments.

SET-PACKING($r$): The restriction where every vertex of $U$ appears in at most $r$
sets from $C$.

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 em-
bodied by a single value $v_a$ and there are $\ell$ 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 $C$.

Consider the trivial situation with only singleton sets in $C$. There we could
simply connect each $x_i$ to each value that occurs as such a singleton. Then a
"packing" of $\ell$ singleton sets in $U$ would obviously give an assignment of $\ell$ 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 colour indicating membership in both sets). If the lefmost value is assigned to some red-green 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 $w, v', w'$ 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.)

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 \( \ell \) sets from \( \mathcal{C} \) into \( U \).

\( \square \)

**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 3-VERTEX-COLORING, which is known to be NP-complete (see for instance[5]).

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 \( \mathcal{X} = A \). 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. \( \square \)

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 \( \Pi \) 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 \( \Pi \) we have \( \alpha \cdot \text{Opt}(\Pi, x) \leq A(x) \leq \text{Opt}(\Pi, x) \), where \( A(x) \) is the output of the algorithm on input \( x \), and \( \text{Opt}(\Pi, x) \) is the value of the optimum solution. Clearly, the larger the *approximation factor* \( \alpha \), the better the quality of approximation. 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 \( \text{APX} \).

### 3.2 Improving the 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 profit \( \text{Opt} \). 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 \( \text{Opt} \geq f(S) \).

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

\[
 f(X_1) + f(X_2) + f(X_1 \oplus X_2) \geq g(X_1) + g(X_2) + g(X_1 \oplus X_2) = 2g(X_1 \cup X_2) = 2 \text{Opt}.
\]

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

For \( k > 2 \), we generalise the notion of symmetric differences appropriately. Define \( X_i := X_i \setminus \bigcup_{k \neq i} X_k \) and consider the reduced pairs \( S_{ij} = X_i \cup Y_j \). As before, we can determine \( f(S_{ij}) \) exactly for any index pair \( i \neq j \) by independent computation of maximum matchings on \( X_i \) and \( X_j \). A careful choice of coefficients yields

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

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

**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 \( \text{Opt} \) 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 non-negative weights \( \zeta_S \), we have
\[
\sum_S \xi_S f(S) \geq \sum_S \xi_S g(S) \geq F_{\xi} \cdot \text{Opt}
\]
where \( F_{\xi} = \min_{x \in X} \sum_{S \in g} \xi_S \); the last inequality holds because each edge \((x, d)\) of the optimal solution \( M \) is counted with a total weight of \( \sum_{S \in g} \xi_S \). By averaging, the largest term \( f(S) \) is at least \( F_{\xi} \cdot \text{Opt} / T_{\xi} \), 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 \( R \) denote the collection of maximal sets \( S \subseteq X \) for which \( f(S) \) can be efficiently computed (and is upper bounded by \( \text{Opt} \)). Also, let \( C \) denote the family of maximal subsets of \( X \) entirely contained in either of each \( X_i \) or \( X \setminus X_i \). That is, every \( C \in C \) is identified with a vector \((v_1, v_2, \ldots, v_k) \in \{0, 1\}^k \setminus \{0^k\} \), such that \( C = \bigcap_{i=1}^k \bigcup_{v_i=0} \bigcup_{X_i} \). Clearly, \( |C| = 2^k - 1 \). We characterise the sets in \( R \) and observe that \( |R| = \mathcal{O}(2k^k) \). Also, we note that for each \( R \in R \) and each \( C \in C \), \( C \) is either completely contained in or completely outside \( R \).

We now wish to compute, for each \( R \in R \), a weight \( \xi_R \) such that the corresponding ratio \( F_{\xi}/T_{\xi} \) is maximised. Define an \( \alpha \times \beta \) 0-1 matrix \( A = (a_{RC})_{R,C} \) where \( a_{RC} = 1 \) iff \( C \subseteq R \). Consider the following pair of primal-dual linear programs:

\[
\begin{align*}
\lambda_p^*(k) &= \max_{\lambda} \lambda \quad &s.t.\quad A^T \xi \geq \lambda e_{\beta} \\
\lambda_D^*(k) &= \min_{\lambda} \lambda \quad &s.t.\quad Az \leq \lambda e_{\alpha} \\
&\quad &e_{\alpha}^T \xi = 1 \\
&\quad &\xi \geq 0
\end{align*}
\]

over \( \xi \in \mathbb{R}^\alpha \) and \( z \in \mathbb{R}^\beta \), where \( e_{\alpha} \) and \( e_{\beta} \) are the vectors of all ones of dimensions \( \alpha \) and \( \beta \) respectively. A feasible solution \( \xi \) to the primal assigns weights (normalised so that \( T_{\xi} = 1 \)) to each \( R \in R \) 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 \( \lambda_p^*(k) = \lambda_D^*(k) = 2/(k + 1) \). There is a primal optimal solution \( \xi \in \mathbb{R}^\alpha \) whose support has size \( \{|R \in [\alpha] : \xi_R > 0\}| = k + (\binom{k}{2})2^{k-2} \).

Due to space limitations, we skip the proof of this theorem; it can be found in the full version of this paper. 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 \( R \in R \) having non-zero \( \xi_R \). 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 $2^k\text{poly}(n, m, d, k)$ 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 $k \geq 2$

Recall that completeness within APX is defined through L reductions, see for instance [16]. So an approximation scheme for an APX-complete problem translates into such a scheme for any problem in APX.

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 $\ell$ of variables $x_i$, we let $\ell = p$, the cardinality of $C$. 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 $3 \geq 3$ other sets.) Thus, if the optimal set packing has $k_{\text{opt}}$ sets then $k_{\text{opt}} \geq p/4$.

Let $s_{\text{opt}}$ denote the value of an optimal solution to the SIM-MATCH instance constructed. Note that $s_{\text{opt}}$ counts variables, while $k_{\text{opt}}$ counts sets. We claim that $s_{\text{opt}} = k_{\text{opt}} + M$. The relation $\geq$ follows simply from assigning the $k_{\text{opt}}$ 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 $\alpha$. That is, we can find in polynomial time a feasible assignment on $s$ variables, where $s$ is at least $\alpha s_{\text{opt}}$. Then $s = k' + M \geq \alpha (k_{\text{opt}} + M)$, so $k' \geq \alpha k_{\text{opt}} - M(1 - \alpha) \geq \alpha k_{\text{opt}} - 8p(1 - \alpha) \geq \alpha k_{\text{opt}} - 8(4k_{\text{opt}})(1 - \alpha) = k_{\text{opt}} (33\alpha - 32)$. Thus, an $\alpha$-approximation for SIM-MATCH gives a $(33\alpha - 32)$-approximation for 3-SET-PACKING(2). This gives the desired L reduction.

Plugging 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 within a factor of $1 - 1/3900$ 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 [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

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

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 $P_I$ that is not half-integral](image)

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. $P_L = P_L$. 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 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\(^5\) that it is in fact a vertex of $P_L$ and is outside $P_T$.

References


\(^5\) The software PORTA [13] was used to find this vertex.