Egerváry Research Group
on Combinatorial Optimization

Technical reportS

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

Matchings under distance constraints II.

Péter Madarasi

November 27, 2023

Matchings under distance constraints II.

Péter Madarasi*

Abstract

This paper introduces the *d-distance b-matching problem*, in which we are given a bipartite graph $G = (S, T; E)$ with $S = \{s_1, \dots, s_n\}$, a weight function on the edges, an integer $d \in \mathbb{Z}_+$ and a degree bound function $b : S \cup T \to \mathbb{Z}_+$. The goal is to find a maximum-weight subset $M \subseteq E$ of the edges satisfying the following two conditions: 1) the degree of each node $v \in S \cup T$ is at most $b(v)$ in M, 2) if $s_i t, s_j t \in M$, then $|i - j| \geq d$. In the cyclic version of the problem, the nodes in S are considered to be in cyclic order. We get back the *(cyclic) d-distance matching problem* when $b(s) = 1$ for $s \in S$ and $b(t) = \infty$ for $t \in T$.

We prove that the d-distance matching problem is APX-hard, even in the unweighted case. We show that $(2 - \frac{1}{d})$ is a tight upper bound on the integrality gap of the natural integer programming model for the cyclic d-distance b-matching problem provided that $(2d - 1)$ divides the size of S. For the non-cyclic case, the integrality gap is shown to be at most $(2 - \frac{2}{d})$. The proofs give approximation algorithms with guarantees matching these bounds, and also improve the best known algorithms for the (cyclic) d-distance matching problem.

In a related problem, our goal is to find a permutation of S maximizing the weight of the optimal d-distance b-matching. This problem can be solved in polynomial time for the (cyclic) d-distance matching problem — even though the (cyclic) d-distance matching problem itself is NP-hard and also hard to approximate arbitrarily. For (cyclic) d-distance b-matchings, however, we prove that finding the best permutation is NP-hard, even if $b \equiv 2$ or $d = 2$, and we give e-approximation algorithms.

*Department of Operations Research, ELTE Eötvös Loránd University and the HUN-REN–ELTE Egerváry Research Group on Combinatorial Optimization, Pázmány Péter sétány 1/C, 1117 Budapest, Hungary. madarasip@staff.elte.hu

1 Introduction

In this paper, we introduce a natural generalization of the *d-distance matching problem* [1], where the degree upper bound function in S can be other than the all-one function, and degree bounds can be posed on the nodes in T as well. Given a bipartite graph $G = (S, T; E)$ with $S = \{s_1, \ldots, s_n\}$, a positive integer $d \in \mathbb{N}$ and a function $b : S \cup T \to \mathbb{Z}_+$, an edge set $M \subseteq E$ is called *d-distance b-matching* if 1) the degree of each node $v \in S \cup T$ is at most $b(v)$ in M and 2) if $s_it, s_jt \in M$ for $i \neq j$, then $|i - j| \geq d$. A d-distance b-matching is called *perfect* if the degree of each node $s \in S$ is exactly $b(s)$. In the *cyclic* version of the problem, the nodes in S are considered to be in a cyclic order, and 2) is required cyclically, that is, if $s_it, s_jt \in M$ for $i \neq j$, then both $|i - j| \geq d$ and $|i - j| \leq n - d$ must hold.

In the (cyclic) d-distance b-matching problem, the goal is to find a maximum-weight (cyclic) d-distance b-matching for a given weight function $w : E \to \mathbb{R}$. The special case $w \equiv 1$ is referred to as the *unweighted* problem. Note that the special case when $b(v) = 1$ if $v \in S$ and ∞ if $v \in T$ is the d-distance matching problem, which was introduced in an earlier article from the same author [1].

The (perfect) d-distance b-matching problem is not only a natural problem extending the literature of matchings, but it also appears in several applications, which are natural extensions of the situations in which the d-distance matching problem could be applied, see [1]. For example, imagine n consecutive all-day events $s_1, \ldots, s_n$, each of which must be assigned $b(s_i)$ of watchmen $t_1, \ldots, t_k$. For each event s_i, a set of possible watchmen is given — those who are qualified to be on guard at event s_i. Appoint exactly $b(s_i)$ watchmen to event s_i such that no watchman is assigned to more than one of any d consecutive events, where $d \in \mathbb{N}$ is given, and each watchman t_j is on guard at most $b(t_j)$ events. In the *weighted* version of the problem, let $w_{s_it_j}$ denote the level of safety of event s_i if watchman t_j is on watch, and the objective is to maximize the level of overall safety.

As another application of the above question, consider n items $s_1, \ldots, s_n$ one after another on a conveyor belt, and k machines $t_1, \ldots, t_k$. Each item s_i is to be processed on the conveyor belt by $b(s_i)$ of the qualified machines $N(s_i) \subseteq \{t_1, \ldots, t_n\}$ such that if a machine processes item s_i, then it cannot process any of the next $(d - 1)$ items — because the conveyor belt is running.

Previous work In the special case $d = |S|$, one gets the classic b-matching problem in bipartite graphs. For $d = 1$, the problem reduces to the b-matching problem in bipartite graphs, and we will see that it is a special case of the circulation problem for $d = 2$.

A feasible d-distance b-matching M can be thought of as a b-matching that does not contain the edge set $\{s_it, s_jt\}$ for any $t \in T$ and $|i - j| \leq d$. A similar problem is the *$K_{p,p}$-free p-matching problem* [2]. Here one is given an arbitrary family $\mathcal{T}$ of the subgraphs of G isomorphic with $K_{p,p}$, and the goal is to find a maximum-cardinality b-matching which does not induce any subgraph in $\mathcal{T}$, where $b : S \cup T \to \{0, \ldots, p\}$. This problem can be solved in polynomial time. Note that in the d-distance b-matching problem, b is arbitrary and the type of the forbidden subgraphs is $K_{2,1}$. Another

similar problem is the following. Given a partition $E_1, \ldots, E_k$ of E and positive integers $r_1, \ldots, r_k$, find a perfect matching M for which $|M \cap E_i| \leq r_i$. The problem is introduced and shown to be NP-complete in [3]. Note that the side constraints in the distance matching problem are similar, but the degree constraints are different and our edge sets do not form a partition of E. Several other versions of the "restricted" (b-)matching problem have been introduced, for example in [4, 5, 6, 7].

The perfect d-distance matching problem is a special case of the list-coloring problem on interval graphs [8] and also of the frequency assignment problem [9], as it was shown in [1].

The d-distance matching problem was shown to be NP-hard and an FPT algorithm parameterized by d was given in [1]. An efficient algorithm was also described for the case when the size of T is a constant. A $(2 - \frac{1}{2d-1})$-approximation algorithm for the weighted d-distance matching problem was given, which also implies that the integrality gap of the natural IP model is at most this value. We also gave a $(3/2+\epsilon)$-approximation algorithm for any constant $\epsilon > 0$ in the unweighted case.

Our results We investigate the integrality gap of the natural IP model and give approximation algorithms for the (cyclic) d-distance b-matching problem in Section 2. In particular, we show that $(2-\frac{1}{d})$ is a tight upper bound on the integrality gap in the cyclic case provided that $(2d-1)$ divides the size of S. Concerning the non-cyclic case, the integrality gap is shown to be at most $(2 - \frac{2}{d})$ for $d \geq 2$. In addition, the proofs provide approximation algorithms with approximation factors matching the bounds above, further improving the algorithms for the (cyclic) d-distance matching problem given in an earlier article [1]. As a special case, this further improves the bound on the integrality gap and the approximation factor for the d-distance matching problem from $(2 - \frac{1}{2d-1})$ to $(2 - \frac{2}{d})$.

Answering an open question from [1], Section 3 proves that the (cyclic) d-distance matching problem is APX-hard, even in the unweighted case.

In Section 4, motivated by the second application mentioned above, a different aspect of the problem is considered, in which our goal is to find a permutation of S maximizing the weight of the optimal d-distance b-matching. We prove that a permutation of S maximizing the weight of the optimal d-distance matching can be found in polynomial time — even though the (cyclic) d-distance matching problem itself is NP-hard and also hard to approximate arbitrarily. For (cyclic) d-distance b-matchings, however, we prove that finding the best permutation is NP-hard, even when $b(s) = 2$ for all $s \in S$ or $d = 2$, and we give e-approximation algorithms for both the cyclic and the non-cyclic cases.

Notation Throughout the paper, assume that $G = (S, T; E)$ is a bipartite graph without loops or parallel edges, unless stated otherwise. Let $\Delta(v)$ and $N(v)$ denote the set of edges incident to node v and the neighbors of v, respectively. For a subset $X \subseteq E$ of the edges, $N_X(v)$ denotes the neighbors of v for edge set X. We use $\deg(v)$ to denote the degree of node v. Let $L_d(s_i)$ and $R_d(s_i)$ denote the nodes in the interval of length (at most) d ending and starting at s_i, respectively, that is,

$L_d(s_i) = \{s_{\max(i-d+1,1)}, \ldots, s_i\}$ and $R_d(s_i) = \{s_i, \ldots, s_{\min(i+d-1,|S|)}\}$. In the cyclic case, $L_d(s_i) = \{s_{i-d+1}, \ldots, s_i\}$ and $R_d(s_i) = \{s_i, \ldots, s_{i+d-1}\}$, where the indices are taken modulo $|S|$. By definition, the minimum and the maximum of the empty set are ∞ and $-\infty$, respectively. Given a function $f : A \to B$, both $f(a)$ and f_a denote the value f assigns to $a \in A$, and let $f(X) = \sum_{a \in X} f(a)$ for $X \subseteq A$. Let χ_Z denote the characteristic vector of set Z, that is, $\chi_Z(y) = 1$ if $y \in Z$, and 0 otherwise. Occasionally, the braces around sets consisting of a single element are omitted, for example $\chi_e = \chi_{\{e\}}$ for $e \in E$. Let $\mathbb{N}$ and $\mathbb{Z}_+$ denote the set of positive and non-negative integers, respectively.

2 Integrality gap and approximation algorithms

In this section, we prove that $(2 - \frac{1}{d})$ is a tight upper bound on the integrality gap of the natural IP model of the d-distance b-matching problem provided that the size of S is divisible by $(2d - 1)$. Then, we show that $(2 - \frac{2}{d})$ is an upper bound on the integrality gap of the non-cyclic version for $d \geq 2$ — without any restriction on the size of S. The proofs also give two approximation algorithms with approximation factors matching the bounds above.

Consider the following LP-relaxation of the natural IP model for the weighted d-distance b-matching problem.

$$\max \sum_{e \in E} w_e x_e \tag{LP1}$$

s.t.

$$x \in \mathbb{R}_+^E \tag{1a}$$

$$\sum_{e \in \Delta(v)} x_e \leq b(v) \qquad \forall v \in S \cup T \tag{1b}$$

$$\sum_{\substack{st \in \Delta(t) \\ s \in R_d(s_i)}} x_{st} \leq 1 \qquad \forall t \in T \; \forall i \in \{1, \ldots, n-d\}. \tag{1c}$$

The LP model for the cyclic case consists of the same conditions but (1c) is required for all $t \in T$ and for all $i \in \{1, \ldots, n\}$. This model will be denoted by (LP1'). When $b(s) = 1$ for all $s \in S$ and $b(t) = \infty$ for all $t \in T$, the integer solutions to (LP1) correspond to the feasible d-distance matchings, and we get back the linear program investigated in [1].

The following theorem gives an upper bound on the integrality gap for the cyclic d-distance b-matching problem.

Theorem 2.1. *If $(2d - 1)$ divides $|S|$, then the integrality gap of (LP1') for the weighted cyclic d-distance b-matching problem is at most $(2 - \frac{1}{d})$, and this bound is tight. Furthermore, there exists a polynomial-time approximation algorithm with the same guarantee.*

Proof. For every $i \in \{1, \dots, 2d-1\}$, let $S_i \subseteq S$ denote the union of the sets $R_d(s_{i+q(2d-1)})$ for $q \in \{0, \dots, \frac{n}{2d-1} - 1\}$. Since the size of S is divisible by $(2d-1)$, the nodes in S_i form intervals of length d in $s_1, \dots, s_n$ and each of these intervals is followed by $(d-1)$ nodes of $S \setminus S_i$ cyclically. For each $i \in \{1, \dots, 2d-1\}$, let $G_i = (S, T; E_i)$ be the subgraph of G on the node set of G whose edge set E_i consists of the edges induced by S_i and T.

First, we prove that the polytope given by (LP1') for G_i is the convex hull of its integer solutions. Observe that constraints (1c) need to be required only for those intervals that are fully included in S_i, because these immediately imply that the constraints hold for the rest of the intervals. The matrix of this reduced linear program is the transpose of the incidence matrices of two laminar families written under each other, which is a well-known network matrix, and the right-hand side is integer, hence the polytope is integer [10, Page 152].

Next, we prove the bound on the integrality gap. Let M_i denote a maximum-weight cyclic d-distance b-matching in G_i, and let M be a maximum-weight solution among $M_1, \dots, M_{2d-1}$. Let $x \in \mathbb{R}_+^E$ be an optimal LP solution for G and let M^* be a maximum-weight d-distance b-matching. It is easy to see that all edges of G appear in exactly d of the graphs $G_1, \dots, G_{2d-1}$, which means that

$$wx = \sum_{e \in E} w_e x_e = \frac{1}{d} \sum_{i=1}^{2d-1} \sum_{e \in E_i} w_e x_e$$

holds. Restricting an optimal LP solution for G to the edge set of G_i, a feasible LP solution is obtained for G_i, so the objective value of this restricted solution can be bounded from above by the LP optimum for G_i, which is equal to the IP optimum $w(M_i)$. From these, one gets that

$$wx \leq \frac{1}{d} \sum_{i=1}^{2d-1} w(M_i) \leq \frac{2d-1}{d} w(M) \leq \frac{2d-1}{d} w(M^*), \tag{2}$$

since M is a feasible integer solution for G. This means that the integrality gap is at most $(2 - \frac{1}{d})$, which was to be proven.

Next, we give a tight example for every $d \in \mathbb{N}$. Let $G = (S, T; E)$ be a complete bipartite graph, where $S = \{s_1, \dots, s_{2d-1}\}$ and $T = \{t\}$. Let $b(s) = 1$ for all $s \in S$, and let $b(t) = \infty$. For $w \equiv 1$, the IP optimum is 1, and $x \equiv \frac{1}{d}$ is an optimal LP solution, meaning that the LP optimum is $(2 - \frac{1}{d})$, hence the bound above is tight.

In fact, this proof shows that M is a $(2 - \frac{1}{d})$-approximate solution, which can be found in polynomial time by solving (LP1') [11] for every graph G_i, therefore we also obtain an approximation algorithm for those instances of the maximum-weight cyclic d-distance b-matching problem in which $(2d-1)|n$, which completes the proof. □

The next theorem improves this upper bound in the non-cyclic case when $b(t) = \infty$ for all $t \in T$. As a special case, this also improves the best known upper bound on the integrality gap for the non-cyclic d-distance matching problem from $(2 - \frac{1}{2d-1})$ [1] to $(2 - \frac{2}{d})$.

Theorem 2.2. *Let $b : S \cup T \to \mathbb{Z}_+$ be such that $b(t) = \infty$ for all $t \in T$, and let $d \geq 2$. The integrality gap of (LP1) for the weighted d-distance b-matching problem is at most $(2 - \frac{2}{d})$. Furthermore, there exists a polynomial-time approximation algorithm with the same guarantee.*

Proof. Let $G = (S, T; E)$ be a bipartite graph, where $S = \{s_1, \ldots, s_n\}$, and let $d \in \mathbb{N}$ and $w : E \to \mathbb{R}_+$. If $(2d-2) \nmid n$, then add $(2d-2-r)$ new isolated nodes to the end of S, where r is such that $0 < r < 2d-2$ and $n = k(2d-2)+r$ for some $k \in \mathbb{Z}_+$. This leaves the feasible d-distance b-matchings in G unchanged, therefore one can assume without loss of generality that $(2d-2)|n$.

We proceed similarly to the first part of the proof of Theorem 2.1, but now we leave out only $(d-2)$ consecutive nodes — instead of $(d-1)$ — between disjoint intervals of length d of S. That is, let $S_i \subseteq S$ denote the union of the sets $R_d(s_{i+q(2d-2)})$ for $q \in \{0, \ldots, \frac{n}{2d-2} - 1\}$, where $i \in \{1, \ldots, 2d-2\}$ and R_d is to be taken in the cyclic sense. Let E_i consist of the edges induced by S_i and T, and let $G_i = (S, T; E_i)$ for $i \in \{1, \ldots, 2d-2\}$. Just as in the proof of Theorem 2.1, we prove that the polytope defined by (LP1) is integer for G_i.

Claim 2.3. *For each $i \in \{1, \ldots, 2d-2\}$, the polytope defined by (LP1) is integer for the subgraph G_i.*

Proof. Without loss of generality, we can assume that G is a complete bipartite graph, and hence the edge set of G_i is the complete graph between S_i and T. Notice that (1b) is required only for $s \in S_i$, and (1c) only for the intervals of S contained in S_i and for the intervals of length d containing the last and the first nodes in two consecutive intervals in S_i, since the rest of the constraints are redundant. It suffices to prove that the matrix of this reduced (LP1) is a network matrix [10, Page 152]. Fix an arbitrary order $t_1, \ldots, t_k$ of the nodes in T. The columns of the matrix of the program, that is the variables, are ordered as follows. Let the columns corresponding to the edges incident to t_j form an interval for all $j \in \{1, \ldots, k\}$, which appear in the order given by $t_1, \ldots, t_k$, and for each t_j, sort the interval of the edges incident to t_j by the index of their endpoint in S. Assume that the rows corresponding to constraints (1c) appear first, in lexicographical order, and then the rows corresponding to constraints (1b) follow, also in lexicographical order. Let L denote the matrix obtained this way, and let B_t denote the submatrix of L given by the edges incident to t and by the constraints (1c) for t. By construction,

$$B_t = \begin{bmatrix} 1 \ldots 1 & & & & & \\ & 1\,1 & & & & \\ & & 1 \ldots 1 & & & \\ & & & 1\,1 & & \\ & & & & \ddots & \\ & & & & & 1\,1 \\ & & & & & 1 \ldots 1 \end{bmatrix},$$

where the zero entries are omitted, and each row contains either two or d one entries. The lines correspond to constraints (1c) alternately for an interval of S_i and for the

interval containing the last and the first node of two consecutive intervals in S_i. For every $t \in T$, there is one such block B_t in L placed diagonally. Furthermore, the rows corresponding to constraints (1b) give k identity matrices side by side, one under each B_t, that is, L looks as follows.

$$L = \begin{bmatrix} \begin{bmatrix} B_{t_1} \end{bmatrix} & & & \\ & \begin{bmatrix} B_{t_2} \end{bmatrix} & & \\ & & \ddots & \\ & & & \begin{bmatrix} B_{t_k} \end{bmatrix} \\ \begin{bmatrix} I \end{bmatrix} & \begin{bmatrix} I \end{bmatrix} & \dots & \begin{bmatrix} I \end{bmatrix} \end{bmatrix}$$

Now, we prove that L is a network matrix. Note that each column of L contains either two or three ones. First, consider the submatrix L' formed by the columns of L containing exactly three ones — we will handle the rest of the columns later. Deleting the full-zero rows from the matrix L', which were created by deleting some of the columns from L, we get that

$$L' = \begin{bmatrix} \begin{bmatrix} A_{t_1} \end{bmatrix} & & & \\ & \begin{bmatrix} A_{t_2} \end{bmatrix} & & \\ & & \ddots & \\ & & & \begin{bmatrix} A_{t_k} \end{bmatrix} \\ \begin{bmatrix} I \end{bmatrix} & \begin{bmatrix} I \end{bmatrix} & \dots & \begin{bmatrix} I \end{bmatrix} \end{bmatrix}, \text{ where } A_t = \begin{bmatrix} 1 & & & \\ 1 & 1 & & \\ & 1 & 1 & \\ & & \ddots & \\ & & & 1 \; 1 \\ & & & 1 \end{bmatrix}$$

for each $t \in T$.

We prove that L' is a network matrix. Denote the size of the identity matrices in the last rows of L' by m. Then each block A_t consists of $(m+1)$ rows and m columns. Let M' denote the submatrix given by the last m rows of L', that is, the identity matrices. Let the tree F be a path P with m (undirected) edges $f_1, \dots, f_m$, which will correspond to the rows of M'. The orientation of these edges will be given later. For each node of the path P, add k new leaves connected to that node. Let $e^i_1, \dots, e^i_k$ denote those newly added leaf edges which are incident to the i^{th} node of P for $i \in \{1, \dots, m+1\}$. Let edge e^i_j correspond to the $((m+1)(j-1)+i)^{\text{th}}$ row of L', in other words, e^i_j belongs to the row containing the i^{th} row of A_{t_j}. Orient the edges of P alternately along the path — the first edge can be oriented arbitrarily, and this determines the direction of the other edges along the path. If the (at most) two arcs of P adjacent to arc e^j_i are oriented towards e^j_i, then we orient e^j_i outwards from the common node. Otherwise, if the (at most) two arcs of P are oriented away from e^j_i, then e^j_i is oriented inwards — since the arcs of P are alternately oriented, only these two cases are possible.

Now, we define the non-tree arcs, which correspond to the columns of the matrix. Each column intersects exactly one of the matrices in the diagonal, say A_{t_j}, and each column contains exactly three non-zero elements. Suppose that the r^{th} column is the i^{th} column of A_{t_j} for some $j \in \{1, \dots, k\}$. Then two of the three ones in the column

are in the i^{th} and $(i+1)^{\text{st}}$ rows of A_{t_j}, to which rows the corresponding arcs are e_j^i and e_j^{i+1}, respectively. The third one is in the j^{th} identity matrix, in the row corresponding to arc f_i. In the tree F, arcs e_j^i, f_i and e_j^{i+1} form a directed path, hence one can add a non-tree arc from the target of this path to its source, which corresponds to the r^{th} column. This shows that L' is a network matrix.

Next, we prove that the original L is also a network matrix by a simple extension of the tree F and the non-tree arcs defined above. Let M denote the submatrix of the last $|E_i|$ rows of L, that is, M consists of the identity matrices of size $|E_i| \times |E_i|$ written side by side.

Observe that if there is exactly one non-zero element in the r^{th} column of B_{t_j} in L for some j, then there is exactly one non-zero element in the r^{th} column of every block $B_{t_{j'}}$ for $j' \in \{1, \dots, k\}$. Each of these columns in L contains exactly two ones. The first one is in the i^{th} row of the corresponding block B_{t_j}, which is associated with arc $e_j^i \in F$. The other one is in the r^{th} row of M. For each $j \in \{1, \dots, k\}$, add a new leaf to F connected to the endpoint of e_j^i belonging to P, and associate it with the r^{th} row of M. Orient it such that it forms a directed path of length two with e_j^i. Finally, add a non-tree arc from the source of this path to its target, which corresponds to the column of L containing the r^{th} column of B_{t_j}. This shows that L is indeed a network matrix. As the right-hand side of (LP1) is integer, we get that the polytope defined by (LP1) is integer for G_i, which completes the proof of the lemma. □

We continue the proof of Theorem 2.2. Let $x \in \mathbb{R}_+^E$ be an optimal LP solution, and let $x^{(i)} \in \mathbb{R}_+^E$ be an optimal LP solution for G_i, where $i \in \{1, \dots, 2d-2\}$. Let M^* be a maximum-weight d-distance b-matching, and let M_i be a maximum-weight d-distance b-matching in G_i, where $i \in \{1, \dots, 2d-2\}$. Chose a maximum-weight solution among $M_1, \dots, M_{2d-2}$, and denote it by M.

Similarly to (2), as each edge of G is contained in exactly d of $G_1, \dots, G_{2d-2}$, and by Claim 2.3, we get that

$$wx \le \frac{1}{d} \sum_{i=1}^{2d-2} \sum_{e \in E_i} w_e x_e^{(i)} = \frac{1}{d} \sum_{i=1}^{2d-2} w(M_i) \le \frac{2d-2}{d} w(M) \le \frac{2d-2}{d} w(M^*),$$

which means that the integrality gap is at most $(2 - \frac{2}{d})$. The proof also shows that the edge set M is a $(2 - \frac{2}{d})$-approximate solution, which can be found in polynomial time, so the proof also gives an approximation algorithm for the maximum-weight d-distance b-matching problem with the same guarantee, provided that $b(t) = \infty$ for all $t \in T$. □

Observe that, for $d = 2$, Claim 2.3 holds for arbitrary b, therefore (LP1) describes the d-distance b-matching polytope.

Remark 2.4. *In the proofs of Theorems 2.1 and 2.2, we constructed a collection of edge sets such that the linear program becomes integer when the problem is restricted to any of them, and every edge is contained in d of the selected edge sets. This means that these two collections of edge sets form so-called* (m, ℓ)-covers *for* $(m, \ell) = (2d-1, d)$

and $(m, \ell) = (2d - 2, d)$*, respectively, which gives an alternative way to finish the proofs, because the existence of an* (m, ℓ)*-cover implies that the integrality gap is at most* $\frac{m}{\ell}$ *[12].*

Remark 2.5. *The local search algorithm given for the unweighted d-distance matching problem [1] also works for the unweighted d-distance b-matching problem. Therefore, we have a* $(3/2 + \varepsilon)$*-approximation algorithm for the latter problem.*

3 Hardness of approximation

In the *double matching problem*, we are given a bipartite graph $G = (S, T; E)$ and two sets $S_1, S_2 \subseteq S$ such that $S_1 \cup S_2 = S$. The goal is to find a maximum weight (size) subset M of the edges for which both $M \cap E_1$ and $M \cap E_2$ are matchings, where E_i denotes the edges induced by S_i and T for $i \in \{1, 2\}$. The double matching problem is known to be APX-hard in the weighted case [12]. This implies that the weighted d-distance matching problem and the unweighted cyclic distance matching problems are APX-hard by a weight-preserving reduction from the double matching problem [1].

In what follows, we prove that the hardness of approximation also applies to the unweighted non-cyclic case.

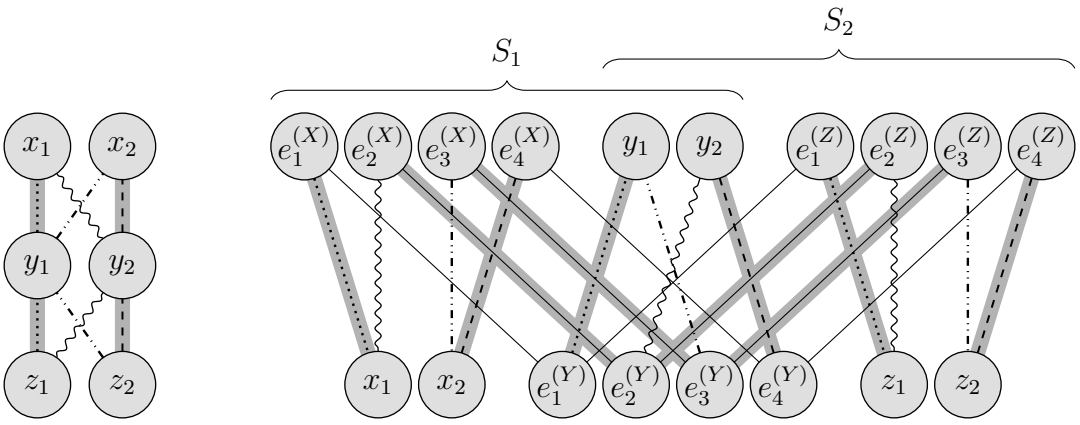
Theorem 3.1. *The unweighted double matching problem is NP-hard to* α*-approximate for any* $\alpha < \frac{950}{949}$.

Proof. Given are three finite disjoint sets X, Y, Z and a set of hyperedges $\mathcal{E} \subseteq X \times Y \times Z$, a subset of the hyperedges $F \subseteq \mathcal{E}$ is called 3-*dimensional matching* if $x_1 \neq x_2, y_1 \neq y_2$ and $z_1 \neq z_2$ for any two distinct triples $(x_1, y_1, z_1), (x_2, y_2, z_2) \in F$. Finding a maximum-size 3-dimensional matching $F \subseteq \mathcal{E}$ cannot be approximated arbitrarily unless P=NP [13]. In fact, the problem remains NP-hard to approximate better than $\frac{95}{94}$ even for 2-*regular* instances, that is, when each element of $X \cup Y \cup Z$ occurs in exactly two triples in $\mathcal{E}$ [14]. To reduce the 2-regular 3-dimensional matching problem to the double matching problem, consider the following construction.

Let $\mathcal{H}_X, \mathcal{H}_Y$ and $\mathcal{H}_Z$ denote three copies of the set of hyperedges $\mathcal{H}$, where the three versions of a hyperedge $e \in \mathcal{H}$ are $e^{(X)} \in \mathcal{H}_X$, $e^{(Y)} \in \mathcal{H}_Y$ and $e^{(Z)} \in \mathcal{H}_Z$. Define a bipartite graph $G = (S, T; E)$, where $S = \mathcal{H}_X \cup Y \cup \mathcal{H}_Z$, $T = X \cup \mathcal{H}_Y \cup Z$ and E is as follows. For each $e \in \mathcal{H}$, add edges $e^{(X)}e^{(Y)}$ and $e^{(Y)}e^{(Z)} \in E$, furthermore, add an edge to G between u and the two hyperedges in $\mathcal{H}_U$ incident to u in H for each $u \in U$, where $U \in \{X, Y, Z\}$. Let $S_1 = \mathcal{H}_X \cup Y$ and $S_2 = Y \cup \mathcal{H}_Z$. For a hyperedge $e = (x, y, z) \in \mathcal{H}$, let $K_e = \{e^{(X)}e^{(Y)}, e^{(Z)}e^{(Y)}, xe^{(X)}, ye^{(Y)}, ze^{(Z)}\} \subseteq E$. Figures 1a and 1b give an example for the construction.

Assume that there exists an α-approximation algorithm for the maximum double matching problem and let M be an α-approximate solution in G. We prove that one can construct a $(\frac{1}{10/\alpha - 9})$-approximate 3-dimensional matching in polynomial time using M, provided that $\alpha < \frac{10}{9}$.

First, consider the following transformation of M. For each $e \in \mathcal{H}$, if $|K_e \cap M| < 3$, then add edges $e^{(X)}e^{(Y)}$ and $e^{(Y)}e^{(Z)}$ to M, and remove all other edges of K_e. After



(a) An instance of the 3-dimensional matching problem.

(b) The corresponding instance of the double matching problem.

Figure 1: Illustration of the proof of Theorem 3.1. Each hyperedge is represented by a unique line style. The highlighted 3-dimensional matching in (a) corresponds to the highlighted solution in (b).

these operations, M remains feasible and its size does not decrease, hence it remains an α-approximate double matching. Observe that after the transformation we have either $|K_e \cap M| = 2$ and hence $K_e \cap M = \{e^{(X)}e^{(Y)}, e^{(Y)}e^{(Z)}\}$, or $|K_e \cap M| = 3$ and hence $K_e \cap M = \{xe^{(X)}, ye^{(Y)}, ze^{(Z)}\}$ for each $e = (x, y, z) \in \mathcal{H}$.

Construct the 3-dimensional matching $F \subseteq \mathcal{H}$ as the set of those hyperedges for which $|K_e \cap M| = 3$. Note that F is feasible, because $K_{e_1} \cap M = \{x_1e_1^{(X)}, y_1e_1^{(Y)}, z_1e_1^{(Z)}\}$ and $K_{e_2} \cap M = \{x_2e_2^{(X)}, y_2e_2^{(Y)}, z_2e_2^{(Z)}\}$ can hold simultaneously only if $x_1 \neq x_2$, $y_1 \neq y_2$ and $z_1 \neq z_2$ — as the degrees of these nodes are at most one in M.

That is, we can construct a 3-dimensional matching F in H such that

$$|M| = 3|F| + 2(2|Z| - |F|) = |F| + 4|Z|,$$

since exactly three edges belong to each hyperedge in F, and two edges belong to each hyperedge in $\mathcal{H} \setminus F$. Applying this for a maximum double matching M^*, we get that

$$|M^*| = |F'| + 4|Z| \leq |F^*| + 4|Z|,$$

where F' denotes the 3-dimensional matching constructed from M^*, and F^* is a maximum 3-dimensional matching in H. Similarly, for any 3-dimensional matching F in H, we can create a double matching M in G such that $|M| = |F| + 4|Z|$, so $|M^*| \geq |F^*| + 4|Z|$, therefore

$$|M^*| = |F^*| + 4|Z|.$$

Hence, for the α-approximate double matching M, the 3-dimensional matching F constructed from M, and for optimal M^* and F^* solutions,

$$|M^*| - |M| = |F^*| - |F| \tag{3}$$

holds.

With the greedy method, we can always construct a 3-dimensional matching of size at least $\frac{|\mathcal{H}|}{4}$, therefore $|F^*| \geq \frac{|\mathcal{H}|}{4} = \frac{|Z|}{2}$. Using that $|S| = 2|\mathcal{H}| + |Z| = 5|Z|$ and that in a double matching M each $s \in S$ has degree at most 1, one gets that $|M^*| \leq 5|Z|$. Therefore,

$$|M^*| \leq 5|Z| \leq 10|F^*| \tag{4}$$

holds. It follows from these observations that

$$\frac{|F|}{|F^*|} = \frac{|F^*| - (|M^*| - |M|)}{|F^*|} = 1 - \frac{|M^*| - |M|}{|F^*|} \geq 1 - 10\frac{|M^*| - |M|}{|M^*|}$$
$$= 1 - 10\left(1 - \frac{|M|}{|M^*|}\right) = 1 - 10\left(1 - \frac{1}{\alpha}\right) = \frac{10}{\alpha} - 9$$

where the first inequality follows from (3), and the second one holds by (4). If $\alpha < \frac{10}{9}$ and $F \neq \emptyset$, then $\frac{|F^*|}{|F|} \leq \frac{1}{10/\alpha - 9}$, so if we had an α-approximate algorithm for the double matching problem, then we could construct a $(\frac{1}{10/\alpha-9})$-approximate 3-dimensional matching in polynomial time. The 2-regular 3-dimensional matching problem is NP-hard to β-approximate for any $\beta < \frac{95}{94}$, which implies that the double matching problem is NP-hard to α-approximate for $\alpha < \frac{950}{949}$. □

By the size-preserving reduction from the double matching problem to the distance matching problem given in [12], the previous theorem implies the following.

Theorem 3.2. *The unweighted distance matching problem is NP-hard to α-approximate for any $\alpha < \frac{950}{949}$.*

Clearly, this result also applies to the more general unweighted cyclic version of the problem. Note that the proof given in [12] gives a slightly larger threshold of $\frac{760}{759}$ in the weighted non-cyclic and in the unweighted cyclic cases.

4 Optimal permutations

This section investigates a slightly different problem, which is motivated by the second application presented in the introduction. It is a natural question whether we can find a permutation of S — which corresponds to the items on the conveyor belt — maximizing the weight of the optimal d-distance b-matchings. Formally, let M^*_π denote an optimal d-distance b-matching under the permutation π of S. We want to find a permutation of S and a d-distance b-matching M^* with respect to this permutation such that $w(M^*) = \max_{\pi \in \mathfrak{S}} w(M^*_\pi)$, where $\mathfrak{S}$ is the set of all permutations of S.

In the next section, a polynomial-time algorithm is described for finding an optimal permutation and an optimal d-distance matching under this permutation (that is, when $b(s) = 1$ for all $s \in S$ and $b(t) = \infty$ for all $t \in T$). Section 4.1, however, proves that the analogous problem is NP-hard for d-distance b-matchings even if $b \equiv 2$ or $d = 2$, and gives e-approximation algorithms for general b in both the cyclic and the non-cyclic cases, where e is Euler's number.

As we have already seen, for a given permutation of S, it is NP-complete to decide whether a perfect (cyclic) d-distance matching exists; and finding a largest one is APX-hard. In this light, it is quite surprising that we can find a permutation of S which maximizes the weight of the maximum-weight (cyclic) d-distance matching — furthermore, an optimal distance matching under the optimal permutation can be found as well.

Before entering the details of this method, we need the following lemma, which is easy to prove by a straightforward reduction to the circulation problem.

Lemma 4.1. *For a bipartite graph $G = (S, T; E)$, a weight function $w : E \to \mathbb{R}_+$ on its edges and integers $k, r \in \mathbb{Z}_+$, we can find a maximum-weight subset of the edges in polynomial time satisfying the following three conditions:*

1) the degrees of the nodes in S are at most 1,

2) the degrees of the nodes in T are at most $(k+1)$, and

3) there are at most r nodes in T with degree exactly $(k+1)$.

Next, we prove that one can find a permutation maximizing the size of the optimal d-distance matching. Note that the first algorithm for the non-cyclic case appeared in [15]. In the rest of this section, a revised, more intuitive approach is presented, which will be modified to handle the cyclic case as well.

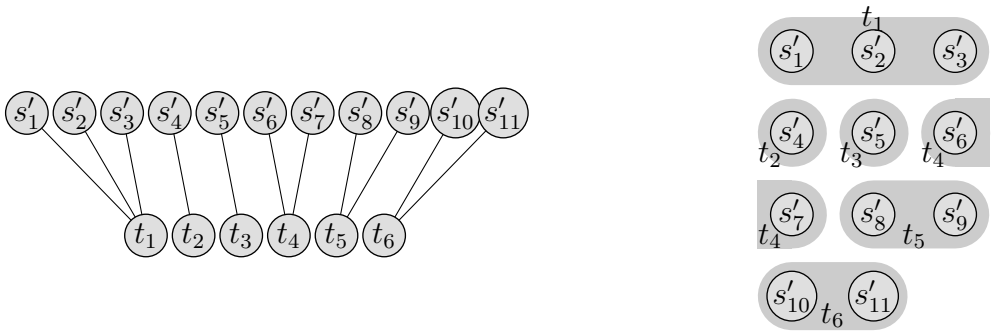
Theorem 4.2. *For a bipartite graph $G = (S, T; E)$, a weight function $w : E \to \mathbb{R}_+$ on its edges and a positive integer $d \in \mathbb{N}$, we can find a permutation of S along with a d-distance matching M in polynomial time such that the weight of M is the largest among all d-distance matchings over all permutations of S.*

Proof. Let $k, r \in \mathbb{Z}_+$ be such that $|S| = kd + r$, where $0 \leq r < d$. Find a maximum-weight edge set M in G such that the degrees of the nodes in S are at most 1, the degrees of the nodes in T are at most $(k+1)$, and there are at most r nodes in T with degree exactly $(k+1)$. Such an edge set M can be found in polynomial time by Lemma 4.1.

Clearly, a maximum-weight d-distance matchings under all permutations fulfill these three conditions, so for this largest possible weight W,

$$w(M) \geq W \tag{5}$$

holds. To show equality, it suffices to construct a permutation of S such that M is a feasible d-distance matching in G. Let $t_1, \ldots, t_{|T|}$ be a permutation of the nodes in T which lists the nodes of degree $(k+1)$ first, then the nodes with degree smaller than k, and finally the nodes of degree k. Let $s'_1, \ldots, s'_n$ be a permutation of S in which the neighbors in M of t_j form an interval for all $j \in \{1, \ldots, |T|\}$ and these intervals appear in the order given by $t_1, \ldots, t_{|T|}$ (the order of the neighbors of any t_j is arbitrary). Figure 2a shows an example for the construction. Now, take a table of size $d \times (k+1)$, and remove all cells from the last column except for the first r. Fill the remaining cells of the table in row-major order with $s'_1, \ldots, s'_n$. Let $s_1 \ldots, s_n$ be



(a) A feasible matching for $d = 4$. The indeces of the nodes correspond to the first step of the construction described in the proof. In this case, $k = 2$ and $r = 3$.

(b) The table corresponding to the graph shown on the left. The optimal permutation is $s'_1, s'_4, s'_7, s'_{10}, s'_2, s'_5, s'_8, s'_{11}, s'_3, s'_6, s'_9$.

Figure 2: Illustration of the proof of Theorem 4.2.

the permutation of S obtained by reading the table in column-major order. Figure 2b shows an example for the table-filling step.

We claim that M is a feasible d-distance matching under the permutation $s_1, \ldots, s_n$. The degrees in M of the nodes in S are clearly at most 1. To see that the distance constraints are met at each node $t \in T$, consider the following three cases. 1) The degree of t in M is $(k + 1)$. This means that the neighbors of t occupy one of the first r rows of the table, which are of length $(k + 1)$. In column-major order, there are $(d - 1)$ other nodes between any two consecutive neighbors of t, which was to be shown. 2) The degree of t in M is smaller than k. The neighbors of t are placed to at most two rows of the table in a row-major manner. Each of these rows is of length k or $(k + 1)$, therefore any two consecutive neighbors in row-major manner have at least $(d - 1)$ other nodes between them. The first and the last neighbors of t have a column between them which contains no neighbors of t, and hence there are more than $(d - 1)$ nodes between them in column-major order. 3) The degree of t in M is k. By construction, the neighbors of t either occupy one of the last $(d - r)$ rows of the table — which are of length k —, or they are placed to at most two rows in row-major manner such that the upper row is of length $(k + 1)$. Similarly to the previous case, there are $(d - 1)$ other nodes between any two consecutive neighbors and also between the first and the last one. These three cases prove that M is feasible under the permutation $s_1, \ldots, s_n$ of S. By (5), this means that M is a heaviest d-distance matching among all distance matchings under all permutations, which completes the proof of the theorem. □

Now we prove the analogous theorem for the cyclic case.

Theorem 4.3. *For a bipartite graph $G = (S, T; E)$, a weight function $w : E \to \mathbb{R}_+$ on its edges and a positive integer $d \in \mathbb{N}$, we can find a permutation of S and a maximum-weight cyclic d-distance matching M with respect to this permutation in polynomial time such that the weight of M is the largest over all permutations of S.*

Proof. We follow the same principle as in the proof of Theorem 4.2. Let $|S| = kd + r$, where $0 \leq r < d$. Find a maximum-weight edge set M in G such that the degrees of the nodes in S are at most 1, and the degrees of the nodes in T are at most k. Such an edge set can be found in polynomial time. Similarly to the proof of Theorem 4.2, it suffices to construct a permutation of S such that M is a feasible d-distance matching in G.

Let $t_1, \ldots, t_{|T|}$ be a permutation of the nodes in T which lists the nodes of degree k first, then the rest of the nodes. Let $s'_1, \ldots, s'_n$ be a permutation of S in which the neighbors in M of t_j form an interval for all $j \in \{1, \ldots, k\}$ and these intervals appear in the order given by $t_1, \ldots, t_{|T|}$ (the order of the neighbors of any t_j is arbitrary). Now, take a table of size $(d+1) \times k$, and remove all cells from the last row except for the first r. Fill the remaining cells of the table in row-major order with $s'_1, \ldots, s'_n$. Let $s_1 \ldots, s_n$ be the permutation of S obtained by reading the table in column-major order. Similarly to the proof for the non-cyclic case, one can prove that M is a feasible d-distance matching under this permutation. □

Observe that Theorems 4.2 and 4.3 extend to the case when degree bounds are also given for the nodes in T. To prove this, one can require that in the initial edge set M, found in the first steps of the proofs, the degree of each node $t \in T$ is also at most $b(t)$ — Lemma 4.1 is easy to modify for finding such an edge set.

4.1 (Cyclic) d-distance b-matchings

This section proves that the analogous problem for d-distance b-matchings is hard, even if $b \equiv 2$ or $d = 2$.

4.1.1 Hardness results

We saw that an optimal permutation can be found in polynomial time for both cyclic and non-cyclic d-distance matchings. This section investigates the complexity of the analogous problem for the d-distance b-matching problem. First, we show that finding an optimal permutation is already hard when $b(s) = 2$ for all $s \in S$ and $b(t) = \infty$ for all $t \in T$, that is, we consider the slight modification of the d-distance matching problem where the degree bound for each node in S is two — instead of the all-one bound.

Theorem 4.4. *It is NP-complete to decide whether there exists a permutation of S such that there is a perfect d-distance b'-matching under this permutation, where $d = |S|/2$ and*

$$b'(v) = \begin{cases} 2 & \text{if } v \in S, \\ \infty & \text{if } v \in T \end{cases} \tag{6}$$

for $v \in S \cup T$.

Proof. In the *C_{4k+2}Free2Factor problem*, a bipartite graph $G' = (S', T'; E')$ is given and the goal is to decide whether it contains a 2-factor (that is, a subgraph in which

the degree of each node is exactly two) such that the length of every cycle in it is a multiple of 4. This problem is known to be NP-complete [16], therefore, it suffices to reduce it to the problem defined in the theorem.

Without loss of generality, we can assume that $|S'| = |T'|$ — otherwise, the instance of the C_{4k+2}Free2Factor problem is not solvable. Let $G = (S, T; E)$ be a copy of G', and add $|S'|$ new nodes to S, and $2|S|$ new nodes to $|T|$. Add $|S'|$ node-disjoint paths of length two on the newly added nodes such that the nodes in the middle of the paths are in S and their endpoints are in T. We show that G' has a 2-factor consisting of cycles whose length is divisible by 4 if and only if there is a permutation of S such that a perfect d-distance b'-matching exists in G, where b' is as defined in (6) and $d = |S|/2$.

Let $s_1, \ldots, s_n$ be a permutation of the nodes in S such that there exists a perfect d-distance b'-matching $M \subseteq E$. The degrees in M of the nodes in S are exactly 2, so M contains all the edges of the paths of length 2 added to G, and the degrees of the nodes in $T' \subseteq T$ are also exactly 2 since $|S'| = |T'|$. Therefore, restricting M to the edge set of the original graph G', we get a 2-factor in G'. We prove that the length of each cycle is a multiple of 4. For all $t \in T$, if $s_i t, s_j t \in M$ for some $i \neq j$ and hence $t \in T'$, then $|i - j| \geq |S'|$. But $|S| = 2|S'|$, thus one of the indices i and j is in $\{1, \ldots, |S|\}$, and the other one is in $\{|S'| + 1, \ldots, 2|S'|\}$. This means that the nodes of S can be divided into two disjoint sets S_1 and S_2 such that one of the neighbors of t is in S_1 and the other one is in S_2 for all $t \in T'$. From this, one gets that every second node of any cycle in M' is in S, and the nodes of the cycle in S are alternately in S_1 and in S_2, so the length of every cycle must be a multiple of 4.

To finish the proof, we show that if there is a 2-factor in G' with cycles whose length is divisible by 4, then there exists a permutation of the nodes in S such that there is a d-distance b'-matching in G. For each cycle of the 2-factor, divide its nodes belonging to S into two sets S_1 and S_2 alternately. Construct a permutation by enumerating the nodes of S_1 in arbitrary order, then the middle nodes of the paths of length two in arbitrary order, and finally the nodes of S_2 in arbitrary order. Under this permutation, the union of the edge set of the 2-factor and the edges of the paths of length two form a d-distance b'-matching, where b' is as defined in (6) and $d = |S|/2 = |S'|$. □

To prove a similar theorem for the cyclic case, we need the following lemma.

Lemma 4.5. *It is NP-complete to decide whether all nodes of a bipartite graph $G = (S, T; E)$ can be covered by node-disjoint cycles of length 4.*

Proof. Given four disjoint sets X_1, X_2, X_3, X_4 and a set of hyperedges $\mathcal{E} \subseteq X_1 \times X_2 \times X_3 \times X_4$, a subset of the hyperedges $F \subseteq \mathcal{E}$ is called 4-*dimensional matching* if $u_1 \neq v_1, u_2 \neq v_2$, $u_3 \neq v_3$ and $u_4 \neq v_4$ for any two distinct hyperedges (u_1, u_2, u_3, u_4), $(v_1, v_2, v_3, v_4) \in F$. It is NP-complete to decide whether there exists a 4-dimensional matching of size $|X_1|$ [17].

We reduce the 4-dimensional matching problem to the problem defined in the lemma statement. Let $H = (X_1 \cup X_2 \cup X_3 \cup X_4, \mathcal{E})$ denote the hypergraph given in the 4-dimensional matching problem, and define an instance of the problem given in the statement of the lemma as follows. Let the node set of G consist of the elements in

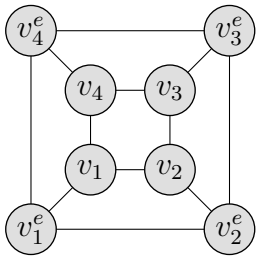


Figure 3: Illustration for the reduction in the proof of Lemma 4.5 for the hyperedge (v_1, v_2, v_3, v_4).

$X_1 \cup X_2 \cup X_3 \cup X_4$ and, for each $e \in \mathcal{E}$, four additional nodes v_1^e, v_2^e, v_3^e and v_4^e. For each hyperedge $(v_1, v_2, v_3, v_4) \in \mathcal{E}$ add the edges v_1v_2, v_2v_3, v_3v_4 and v_4v_1 to G. Also add the edges $v_1^ev_2^e$, $v_2^ev_3^e$, $v_3^ev_4^e$, $v_4^ev_1^e$, and $v_1v_1^e$, $v_2v_2^e$, $v_3v_3^e$, $v_4v_4^e$ to G for all $e \in \mathcal{E}$. Figure 3 illustrates the construction for the hyperedge $(v_1, v_2, v_3, v_4) \in \mathcal{E}$. It is easy to see that G is bipartite: let S consist of the nodes in $X_2 \cup X_4$ and also the nodes v^e for $v \in X_1 \cup X_3$, $e \in \mathcal{E}$, and let T consist of the rest of the nodes. By the definition of E, neither of the two parts induces any edges. We need to show that there is a perfect 4-dimensional matching in H if and only if G has a vertex cover with node-disjoint 4-cycles.

Firstly, given a perfect 4-dimensional matching F in H, we construct the set of 4-cycles as follows. For any hyperedge $e = (v_1, v_2, v_3, v_4) \in F$, select the cycle formed by the nodes v_1, v_1^e, v_2^e, v_2 and the cycle on nodes v_3, v_3^e, v_4^e, v_4. For each hyperedge $(u_1, u_2, u_3, u_4) \in \mathcal{E} \setminus F$, also add the cycle formed by the nodes $u_1^e, u_2^e, u_3^e, u_4^e$. The 4-cycles obtained this way are pairwise node-disjoint, because the hyperedges in F form a 4-dimensional matching. Clearly, they cover all the nodes of G, because F is a perfect 4-dimensional matching.

Secondly, assume that there exists a cover with node-disjoint 4-cycles in G. We modify the set of the cycles as follows. If for a hyperedge $e = (v_1, v_2, v_3, v_4)$ both of the cycles formed by nodes v_1, v_2, v_3, v_4 and $v_1^e, v_2^e, v_3^e, v_4^e$ are in the subgraph, then delete these two cycles and add the cycles formed by the nodes v_1, v_1^e, v_2^e, v_2 and v_3, v_3^e, v_4^e, v_4 instead. Clearly, the new cycles are node-disjoint and cover every node of G. Now, construct a 4-dimensional matching F in H by adding the hyperedge $e = (v_1, v_2, v_3, v_4)$ to F if and only if the cycle formed by the nodes $v_1^e, v_2^e, v_3^e, v_4^e$ is not in the set of the selected 4-cycles. This is a perfect 4-dimensional matching, because if the cycle on nodes $v_1^e, v_2^e, v_3^e, v_4^e$ is not selected, then the two cycles covering these four nodes cover the nodes v_1, v_2, v_3, v_4, and because all nodes of G are in exactly one of the selected 4-cycles. □

Now we are ready to prove the analogous theorem for the cyclic case.

Theorem 4.6. *It is NP-complete to decide whether there exists a permutation of the nodes in S such that there is a perfect cyclic d-distance b'-matching under this permutation, where $d = |S|/2$ and*

$$b'(v) = \begin{cases} 2 & \text{if } v \in S, \\ \infty & \text{if } v \in T \end{cases} \tag{7}$$

for $v \in S \cup T$.

Proof. Clearly, the problem is in NP. We reduce the problem defined in Lemma 4.5 to the problem in the theorem. Let $G = (S, T; E)$ be a bipartite graph, and let $d = |S|/2$. Assume that $|S| = |T|$. We show that the nodes of G can be covered by node-disjoint cycles of length 4 if and only if there exists a permutation of the nodes in S under which a perfect cyclic d-distance b'-matching exists, where b' is as defined in (7). This implies the statement of the theorem, since the former problem is NP-complete by Lemma 4.5.

Firstly, assume that we have a permutation $s_1, \ldots, s_n$ of S under which a perfect d-distance b'-matching M exists. Since the M-degree of every node in S is exactly 2, the size of M is $2|S|$. Furthermore, $d = |S|/2$, therefore the degree of t in M is at most 2. From this, the degree of t in M is exactly 2 by $|S| = |T|$, therefore M is the node-disjoint union of cycles. If one of the neighbors of a node $s \in S$ is $t \in T$, then t has a uniquely defined other neighbor, since there exists exactly one node in S whose cyclic distance is at least $|S|/2$ from s. The same holds for the other neighbor of s. So, for every $s \in S$, the two neighbors of it have the same two neighbors, which gives a cycle of length 4. Hence the edges of the cyclic d-distance b'-matching cover all nodes and is the union of node-disjoint 4-cycles.

Secondly, if there is a set of node-disjoint cycles of length 4 covering the nodes of G, then we can construct a proper permutation of S as follows. For each cycle, take one of its nodes in S, and put them in the first $|S|/2$ positions of the permutation in arbitrary order. This clearly determines the order of the rest of the nodes in S, since for any node s from the first $|S|/2$ nodes, there is exactly one other node in S having a common neighbor with s, so this must be placed exactly $|S|/2$ positions after s. □

Note that the proofs of Theorems 4.4 and 4.6 also show that the problem is NP-complete when $b(s) = b(t) = 2$ for all $s \in S$ and $t \in T$ both in the cyclic and the non-cyclic case. Next, Theorems 4.4 and 4.6 are extended to the case when $b(s) \geq 2$ for all $s \in S$ — instead of $b(s) = 2$.

Theorem 4.7. *Both in the cyclic and the non-cyclic case, it is NP-complete to decide whether there exists a permutation of the nodes in S such that there is a perfect (cyclic) d-distance b''-matching under this permutation, where $b''(s) \geq 2$ for all $s \in S$ and $b''(t) = \infty$ for all $t \in T$.*

Proof. In both cases, the problem is in NP. We prove the theorem for the cyclic and the non-cyclic versions simultaneously by showing that finding a permutation under which a perfect (cyclic) d-distance b''-matching exists is more general than finding a permutation under which a perfect (cyclic) d-distance b'-matching exists, where b' is as defined in (7).

We modify the input graph of the perfect (cyclic) d-distance b'-matching problem as follows. For each $s \in S$, add $(b''(s) - 2)$ new nodes to T, and connect them to s. Let F denote the set of the newly added edges. We prove that there is a permutation of S under which a perfect (cyclic) d-distance b'-matching exists in the original graph if and only if there is a perfect (cyclic) d-distance b''-matching in the new graph under the same permutation of S.

Firstly, if there is a perfect (cyclic) d-distance b'-matching M' in the original graph under some permutation of S, then $M' \cup F$ is a perfect (cyclic) d-distance b''-matching in the new graph under the same permutation.

Secondly, for the reverse direction, let M'' denote a perfect (cyclic) d-distance b''-matching in the new graph under some permutation of S. Take the edge set $M'' \setminus F$, and remove all except two of the edges incident to each node $s \in S$. This way one gets a feasible perfect (cyclic) d-distance b'-matching in the original graph under the same permutation.

This completes the proof both in the cyclic and the non-cyclic version by Theorems 4.4 and 4.6, respectively. □

4.2 (Cyclic) d-distance b-matchings for small d

By Theorems 4.4 and 4.6, finding an optimal permutation is hard when $d = |S|/2$. We show that the problem is also hard for any $d \geq 2$ that is polynomially smaller than $|S|$, which, as a special case, implies that even the case $d = 2$ is NP-complete.

Theorem 4.8. *It is NP-complete to decide whether there exists a permutation of the nodes in S such that there is a perfect (cyclic) d-distance b'-matching under this permutation, where $|S| = d(\ell + 1) - 2$, $d \geq 2$, $\ell = \Omega(|S|^c)$ for some constant $c > 0$, and*

$$b'(v) = \begin{cases} \deg(v) & \text{if } v \in S, \\ \infty & \text{if } v \in T \end{cases} \tag{8}$$

for $v \in S \cup T$.

Proof. The problem is clearly in NP. We give a reduction from the *Hamiltonian path problem* in an undirected simple graph $G' = (V', E')$ to the non-cyclic case when $d = 2$, then we show that the problem stated in the theorem includes this as a special case. Construct a bipartite graph $G = (S, T; E)$ such that $S = V'$ and T is the edge set of the complement of G'. For each $t = uv \in T$, add edges ut and vt to G. We prove that there exists a Hamiltonian path in G' if and only if there is a permutation of the nodes in S under which a perfect d-distance b'-matching exists, where $d = 2$ and b' is as defined in (8).

Firstly, assume that M is a perfect d-distance b'-matching in G under the permutation $s_1, \ldots, s_n$ of S. By the definition of b', this means that $M = E$, therefore $\{s_i t, s_{i+1} t\} \not\subseteq E$ for all $i \in \{1, \ldots, n-1\}$ and for all $t \in T$. But then $s_i s_{i+1} \in E'$ for all $i \in \{1, \ldots, n-1\}$, which means that $s_1, \ldots, s_n$ defines a Hamiltonian path in G'.

Secondly, assume that a Hamiltonian path in G' traverses the nodes of G' in the order $s_1, \ldots, s_n$. This means that there is an edge e_i in E' between s_i and s_{i+1} for all $i \in \{1, \ldots, n-1\}$, that is, $\{s_i t, s_{i+1} t\} \not\subseteq E$ for all $i \in \{1, \ldots, n-1\}$ and for all $t \in T$. This means that $M = E$ is a perfect d-distance b'-matching in G under the permutation $s_1, \ldots, s_n$ of S, where $d = 2$ and b' is as defined above. This completes the proof for $d = 2$ in the non-cyclic case.

Now we prove that if $n = |S| = d(\ell + 1) - 2$ for $\ell = \Omega(n^c)$ and $d \geq 2$, then the non-cyclic problem includes the case $d = 2$ for $G' = (S', T'; E')$ with $|S'| = 2\ell$. Let $G = (S, T; E)$ be a copy of G', and add a new node $\tilde{s}_{q,r}$ to S for all $q \in \{1, \ldots, \ell + 1\}$ and $r \in \{1, \ldots, d - 2\}$. For $r \in \{1, \ldots, d - 2\}$, also add a new node t_r to T and all edges between t_r and $\{\tilde{s}_{r,1}, \ldots, \tilde{s}_{r,\ell+1}\}$.

Firstly, we show that if there is a permutation $s_1, \ldots, s_n$ of S under which E is feasible in G, then there is a permutation of S' under which E' is feasible in G'. We claim that $s_{id-1}, s_{id} \in S'$ for all $i \in \{1, \ldots, \ell\}$. By contradiction, suppose that $s_{id-1} = \tilde{s}_{q,r}$ or $s_{id} = \tilde{s}_{q,r}$ for some q and r. Let t_r denote the only neighbor of $\tilde{s}_{q,r}$. As E is feasible, all neighbors of t_r on the left of $\tilde{s}_{q,r}$ are among the nodes $s_1, \ldots, s_{id-d}$. Similarly, all neighbors of t_r on the right of $\tilde{s}_{q,r}$ are among the nodes $s_{id-1+d}, \ldots, s_n$. Therefore, the degree of t must be at most

$$\left\lceil \frac{id - d}{d} \right\rceil + 1 + \left\lceil \frac{n - (id - 1 + d) + 1}{d} \right\rceil = \ell,$$

which contradicts the fact that the degree of t_r is $(\ell+1)$ by the construction of G. This means that $s_{id-1}, s_{id} \in S'$ for all $i \in \{1, \ldots, \ell\}$, as we claimed. Since the size of S' is 2ℓ, the nodes s_{id-1}, s_{id} for $i \in \{1, \ldots, \ell\}$ are exactly the nodes in S'. Therefore, every interval of length d contains two nodes in S', which means that E' is a feasible solution for G' and $d = 2$ under the permutation of S' obtained by restricting $s_1, \ldots, s_n$ to S'.

Secondly, we prove that if there is a permutation $s'_1, \ldots, s'_{|S'|}$ of S' under which E' is feasible, then there is a permutation of S under which E is feasible. Concatenate the subsequences $\tilde{s}_{q,1}, \ldots, \tilde{s}_{q,d-2}$ for $q = 1, \ldots, \ell + 1$. Then, insert s'_{2q-1} and s'_{2q} in this order right after $\tilde{s}_{q,d-2}$ for $q \in \{1, \ldots, \ell\}$. We claim that E is feasible under the permutation obtained this way. As an interval of length d includes exactly two nodes in S' and these nodes appear in the order given by $s'_1, \ldots, s'_{|S'|}$, the edge set E' is feasible. For each $r \in \{1, \ldots, d - 2\}$, the neighbors of t_r are the nodes $\tilde{s}_{1,r}, \ldots, \tilde{s}_{\ell+1,r}$, which do not appear in an interval of length d in the permutation defined above. This means that the edge set $E \setminus E'$ is feasible as well. Since E' and $E \setminus E'$ are node-disjoint, this implies that E is also feasible, which was to be shown. This completes the proof in the non-cyclic case.

To show the hardness of the cyclic case, one can give a reduction from the *Hamiltonian cycle problem*. The proof is a straightforward modification of the reduction for the non-cyclic case, therefore it is left to the reader. □

In the previous theorem, we did not assume that the coordinates of b are small, unlike in Theorems 4.4 and 4.6. It remains open whether the problem becomes tractable when $d = 2$ and the coordinates of b are small, for example $b \equiv 2$.

4.3 Approximation algorithms for finding a permutation

In this section, we give e-approximation algorithms for finding the best (cyclic) permutation under which the weight of the optimal (cyclic) d-distance b-matching is as large as possible. The approximation algorithms also give an e-approximate (cyclic)

Algorithm 1 Randomized approximation algorithm for cyclic S-permutations

For $v \in S \cup T$, let $b'(v) = \begin{cases} b(v) & \text{if } v \in S, \\ \min\{\lfloor \frac{n}{d} \rfloor, b(t)\} & \text{if } v \in T. \end{cases}$
Find a maximum-weight b'-matching $\widehat{M}$ in G.
Generate a cyclic permutation $s_1, \ldots, s_n$ of S uniformly at random.
$M := \emptyset$
for $t \in T$ **do**
  for $i = 1, \ldots, n$ **do**
    if $s_i t \in \widehat{M}$ and $s_{i-d+1}t, \ldots, s_{i-1}t \notin \widehat{M}$ **then**
      $M := M \cup \{s_i t\}$
output $s_1, \ldots, s_n$ and M

d-distance b-matching under the permutation. Both algorithms are randomized, but they are easy to de-randomize, which we briefly discuss at the end of the section.

First, consider the cyclic version of the problem. Algorithm 1 finds a maximum-weight b'-matching $\widehat{M}$ in G for the b' defined in the algorithm and takes a random cyclic permutation of S. Then, for each $t \in T$, it adds an edge $s_i t \in \widehat{M}$ to the solution if and only if t and the $(d-1)$ nodes cyclically before s_i induce no edges in $\widehat{M}$. The algorithm returns the chosen permutation and the union of the selected edges. Clearly, this edge set is a feasible cyclic d-distance b-matching under the chosen permutation, because 1) the degree of any node $s \in S$ is at most $b(s)$, since the found edge set is a subset of $\widehat{M}$, and 2) the distance constraints are met at each node $t \in T$, since no edge st is added for which an edge $s't \in \widehat{M}$ exists such that s' is one of the $(d-1)$ nodes before s cyclically.

The following theorem gives a lower bound on the expected weight of the solution found by Algorithm 1.

Theorem 4.9. *Algorithm 1 outputs a cyclic permutation $s_1, \ldots, s_n$ of S and a feasible cyclic d-distance b-matching M whose expected weight is at least*

$$\max\left\{\left(1-\frac{1}{d}\right)^{d-1}, \left(1-\frac{1}{k}\right)^{k-1}\right\} \tag{9}$$

times the weight of the heaviest cyclic d-distance b-matching under all permutations, where $k = \max_{t \in T} b'(t)$ for the b' defined in Algorithm 1. This lower bound is tight.

Proof. As we have already seen, Algorithm 1 returns a feasible solution under the randomly chosen permutation. Let $\mathbb{E}(n, d, k)$ denote the expected weight of the solution found by the algorithm, and let $\alpha(d, k)$ denote the lower bound given by (9). First, we consider the unweighted case when $T = \{t\}$. Without loss of generality, one can assume that $k \geq 2$ and $d \leq n$. Note that $n \geq dk$ holds by the definition of b'. Let $P(n, d, k)$ be the probability that a given edge is added to the solution. By definition,

$$P(n, d, k) = \prod_{i=1}^{d-1} \frac{n-k-(i-1)}{n-i}. \tag{10}$$

Clearly, $\mathbb{E}(n,d,k) = kP(n,d,k)$. Observe that

$$P(n,d,k) \geq P(dk,d,k) \tag{11}$$

holds for all $n,d,k \in \mathbb{N}$ provided that $n \geq dk$, because dk is the smallest possible size of S when the degree of t can be k in $\widehat{M}$, and if the number of nodes in S is larger than dk, then all additional nodes must be isolated, hence the probability of an edge being added can be only larger. The value of $P(dk,d,k)$ can be expressed as follows.

$$P(dk,d,k) = \prod_{i=1}^{d-1} \frac{kd-k-i+1}{kd-i} = \frac{\prod_{j=\max\{1,d-k+1\}}^{d-1} kd-k-j+1}{\prod_{i=1}^{\min\{k-1,d-1\}} kd-i}$$
$$= \prod_{i=1}^{\min\{k-1,d-1\}} \frac{kd-k-(d-i)+1}{kd-i} = \prod_{i=1}^{\min\{k-1,d-1\}} \frac{(k-1)d-k+i+1}{kd-i},$$

where the first equation holds by (10), and the third one because the product is telescopic. From this, we immediately get that $P(dk,d,k)$ is monotone decreasing in d for all $k \in \mathbb{N}$, and it tends to $\left(1-\frac{1}{k}\right)^{k-1}$ as d goes to infinity for all $k \in \mathbb{N}$. This implies that

$$P(dk,d,k) \geq \left(1-\frac{1}{k}\right)^{k-1} \tag{12}$$

holds for all $d,k \in \mathbb{N}$. Similarly, $P(dk,d,k)$ is monotone decreasing in k for all $d \in \mathbb{N}$, and it tends to $\left(1-\frac{1}{d}\right)^{d-1}$ as k goes to infinity for all $k \in \mathbb{N}$. This implies that

$$P(dk,d,k) \geq \left(1-\frac{1}{d}\right)^{d-1} \tag{13}$$

holds for all $d,k \in \mathbb{N}$.

By (11), (12) and (13), $P(n,d,k) \geq \alpha(d,k)$ holds for all $n,d,k \in \mathbb{N}$ provided that $n \geq dk$, which completes the proof of the unweighted case when $|T| = 1$. The bounds given by (12) and (13) are (asymptotically) tight, therefore (9) cannot be improved, as we stated in the theorem.

Since all edges incident to t appear in the output of Algorithm 1 with equal probability, the expected weight of the returned edges is at least $\alpha(d,k)w(\widehat{M} \cap \Delta(t))$, which was to be shown in the weighted case when $|T| = 1$.

We continue with the general weighted case, that is, when the size of T is arbitrary. Let $\mathfrak{S}_C$ denote the set of all cyclic permutations of S, and let $M_\pi \subseteq \widehat{M}$ denote the feasible cyclic d-distance b-matching returned by the algorithm when it selects the cyclic permutation $\pi \in \mathfrak{S}_C$. The following computation leads to the bound stated in

the theorem.

$$\begin{aligned}\mathbb{E}(n,d,k) &= \frac{\sum_{\pi\in\mathfrak{S}_C} w(M_\pi)}{|\mathfrak{S}_C|} = \frac{\sum_{\pi\in\mathfrak{S}_C}\sum_{t\in T} w(M_\pi\cap\Delta(t))}{|\mathfrak{S}_C|}\\ &= \sum_{t\in T}\frac{\sum_{\pi\in\mathfrak{S}_C} w(M_\pi\cap\Delta(t))}{|\mathfrak{S}_C|} = \sum_{t\in T} P(n,d,\deg_{\widehat{M}}(t))w(\widehat{M}\cap\Delta(t))\\ &\geq \sum_{t\in T}\alpha(d,k)w(\widehat{M}\cap\Delta(t)) = \alpha(d,k)w(\widehat{M}) \geq \alpha(d,k)w(M^*),\end{aligned}$$

where M^* is an optimal cyclic d-distance b-matching under all permutations. The first inequality holds by the case $|T| = 1$, and the second one because any cyclic d-distance b-matching must respect the degree bounds posed by b' and $\widehat{M}$ is a heaviest b'-matching for the b' defined in Algorithm 1. This completes the proof of the theorem. □

It is well known that (9) is monotone decreasing and tends to $\frac{1}{e}$ as k and d go to infinity. This immediately implies the following.

Theorem 4.10. *Algorithm 1 outputs a cyclic permutation $s_1,\dots,s_n$ of S and a feasible cyclic d-distance b-matching whose expected weight is at least $\frac{1}{e}$ times the weight of the heaviest cyclic d-distance matching under all permutations. This lower bound is tight.*

Now, we turn to the non-cyclic case.

Algorithm 2 Randomized approximation algorithm for S-permutations

For $v \in S\cup T$, let $b'(v) = \begin{cases} b(v) & \text{if } v\in S,\\ \min\{\lceil\frac{n}{d}\rceil, b(t)\} & \text{if } v\in T.\end{cases}$
Find a maximum-weight b'-matching $\widehat{M}$ in G.
Generate a permutation $s_1,\dots,s_n$ of S uniformly at random.
$M := \emptyset$
for $t \in T$ **do**
  for $i = 1,\dots,n$ **do**
    if $s_i t \in \widehat{M}$ and $s_{\max\{1,i-d+1\}}t,\dots,s_{i-1}t \notin \widehat{M}$ **then**
      $M := M\cup\{s_i t\}$
output $s_1,\dots,s_n$ and M

Algorithm 2 is the analog of Algorithm 1 for the non-cyclic d-distance b-matching problem. First, the algorithm finds a maximum-weight b'-matching $\widehat{M}$ in G for the b' defined in Algorithm 2, and takes a random permutation of S. Then, for each $t \in T$, it selects an edge $s_i t \in \widehat{M}$ if t and the (at most) $(d-1)$ nodes before s_i induce no edges in $\widehat{M}$. It returns the chosen permutation and the union of the selected edges. Similarly to Algorithm 1, the edge set returned by the algorithm is a feasible d-distance b-matching under the chosen permutation.

The following theorem for the non-cyclic version is analogous to Theorem 4.9.

Theorem 4.11. *Algorithm 2 outputs a permutation $s_1, \dots, s_n$ of S and a feasible d-distance b-matching whose expected weight is at least*

$$\max\left\{\left(1-\frac{1}{d}\right)^{d-1}, \frac{1+(k-1)\left(1-\frac{1}{k-1}\right)^{k}}{k}\right\} \tag{14}$$

times the weight of the heaviest d-distance b-matching under all permutations, where $k = \max_{t\in T} b'(t)$ for the b' defined in Algorithm 2. This lower bound is tight.

Proof. The outline of the proof is similar to that of Theorem 4.9, but the technical details are slightly more complicated. Let $\mathbb{E}(n,d,k)$ denote the expected weight of the solution returned by the algorithm, and let $\beta(d,k)$ denote the lower bound given by (14). Similarly to the proof of Theorem 4.9, consider the case when $T = \{t\}$ and the problem is unweighted. Without loss of generality, one can assume that $k \geq 2$ and $d \leq n$. Let $P(n,d,k)$ be the probability that a given edge is added to the solution, and let $P(n,d,k,i)$ denote the probability that the edge incident to s_i is added to the solution. By definition,

$$P(n,d,k) = \frac{1}{n}\sum_{i=1}^{n} P(n,d,k,i),$$

and

$$P(n,d,k,i) = \prod_{j=1}^{\min\{d-1,i-1\}} \frac{n-k-(j-1)}{n-j}.$$

Clearly, $\mathbb{E}(n,d,k) = kP(n,d,k)$. Observe that

$$P(n,d,k) \geq P((d-1)k+1,d,k)$$

holds for all $n,d,k \in \mathbb{N}$ provided that $n \geq (d-1)k+1$, because $((d-1)k+1)$ is the smallest possible size of S when the degree of t can be k in $\widehat{M}$, and if the number of nodes in S is larger, then all the extra nodes are isolated, hence the probability that an edge is added can be only larger. Let $\bar{n} = (d-1)k+1$. The value of $P(\bar{n},d,k)$

can be expressed as follows.

$$
\begin{aligned}
P(\bar{n},d,k) &= \frac{1}{\bar{n}}\sum_{i=1}^{\bar{n}} P(\bar{n},d,k,i) = \frac{1}{\bar{n}}\sum_{i=1}^{\bar{n}}\prod_{j=1}^{\min\{d-1,i-1\}}\frac{\bar{n}-k-(j-1)}{\bar{n}-j} \\
&= \frac{1}{\bar{n}}\sum_{i=1}^{d-1}\prod_{j=1}^{i-1}\frac{\bar{n}-k-(j-1)}{\bar{n}-j} + \frac{\bar{n}-d+1}{\bar{n}}\prod_{j=1}^{d-1}\frac{\bar{n}-k-(j-1)}{\bar{n}-j} \\
&= \frac{1}{(k-1)d+1}\sum_{i=1}^{d-1}\prod_{j=1}^{i-1}\frac{(k-1)d-k-j+2}{(k-1)d+1-j} \\
&\quad + \frac{(k-1)d-d+2}{(k-1)d+1}\prod_{j=1}^{d-1}\frac{(k-1)d-k-j+2}{(k-1)d+1-j} \\
&= \frac{1}{(k-1)d+1}\sum_{i=1}^{d-1}\prod_{j=1}^{\min\{i-1,k-1\}}\frac{(k-1)d+j-i-k+2}{(k-1)d-j+1} \\
&\quad + \frac{(k-2)d+2}{(k-1)d+1}\prod_{j=1}^{\min\{k-1,d-1\}}\frac{(k-2)d-k+j+2}{(k-1)d-j+1}, \qquad (15)
\end{aligned}
$$

where the last equation holds by rearranging the products. Let $f(d,k)$ and $g(d,k)$ denote the first and the second summand in the right hand-side of (15), respectively. Clearly,

$$
\lim_{d\to\infty} g(d,k) = \lim_{d\to\infty}\frac{(k-2)d+2}{(k-1)d+1}\prod_{j=1}^{\min\{k-1,d-1\}}\frac{(k-2)d-k+j+2}{(k-1)d-j+1} = \left(\frac{k-2}{k-1}\right)^k. \qquad (16)
$$

To derive the limit of $f(d,k)$, we need the following computation.

$$
\begin{aligned}
((k-1)d+1)f(d,k) &= \sum_{i=1}^{d-1}\prod_{j=1}^{\min\{i-1,k-1\}}\frac{(k-1)d+j-i-k+2}{(k-1)d-j+1} \\
&= \sum_{i=1}^{d-1}\prod_{j=1}^{\min\{i-1,k-1\}}\frac{(k-1)d-k-j+2}{(k-1)d-j+1} = \frac{\sum_{i=1}^{d-1}\binom{(k-1)d-i+1}{k-1}}{\binom{(k-1)d}{k-1}} \\
&= \frac{\binom{(k-1)d+1}{k}-\binom{(k-2)d+2}{k}}{\binom{(k-1)d}{k-1}} = \frac{((k-1)d+1)\binom{(k-1)d}{k-1}-((k-2)d+2)\binom{(k-2)d+1}{k-1}}{k\binom{(k-1)d}{k-1}} \\
&= \frac{(k-1)d+1}{k} - \frac{((k-2)d+2)\binom{(k-2)d+1}{k-1}}{k\binom{(k-1)d}{k-1}} \\
&= \frac{(k-1)d+1}{k} - \frac{(k-2)d+2}{k}\prod_{j=1}^{k-1}\frac{(k-2)d-j+2}{(k-1)d-j+1}, \qquad (17)
\end{aligned}
$$

where the first equation holds by the definition of f, the second one by rearranging the product, and the fourth one by applying the binomial identity $\sum_{q=0}^{N}\binom{q}{K} = \binom{N+1}{K+1}$

twice. Using (17), we get that

$$\lim_{d\to\infty} f(d,k) = \lim_{d\to\infty} \frac{1}{k} - \frac{(k-2)d+2}{k((k-1)d+1)} \prod_{j=1}^{k-1} \frac{(k-2)d-j+2}{(k-1)d-j+1} = \frac{1-\left(\frac{k-2}{k-1}\right)^k}{k} \quad (18)$$

for all $k \in \mathbb{N}$. By (15), (16) and (18), $P(\bar{n}, d, k)$ tends to

$$\left(\frac{k-2}{k-1}\right)^k + \frac{1-\left(\frac{k-2}{k-1}\right)^k}{k} = \frac{1+(k-1)\left(\frac{k-2}{k-1}\right)^k}{k}$$

as d goes to infinity for all $k \geq 2$. Observe that $P(n, d, k, i)$ is non-increasing in d, and therefore so is $P(n, d, k)$ for all $n, k \in \mathbb{N}$. This implies that

$$P(n,d,k) \geq \frac{1+(k-1)\left(\frac{k-2}{k-1}\right)^k}{k} \quad (19)$$

holds for all $n, d, k \in \mathbb{N}$ provided that $n \geq (k-1)d+1$. Similarly, $P(n, d, k)$ is non-increasing in k for all $n, d \in \mathbb{N}$. From (15), it is easy to see that $P(\bar{n}, d, k)$ tends to $\left(1-\frac{1}{d}\right)^{d-1}$ as k goes to infinity, therefore,

$$P(n,d,k) \geq \left(1-\frac{1}{d}\right)^{d-1} \quad (20)$$

holds. By (19) and (20), $P(n, d, k) \geq \alpha(d, k)$ follows for all $n, d, k \in \mathbb{N}$ provided that $n \geq (k-1)d+1$, which completes the proof of the unweighted case when $|T| = 1$. The bounds given in (19) and (20) are (asymptotically) tight, therefore (14) cannot be improved, as we stated in the theorem.

Since all edges incident to t appear in the output of Algorithm 2 with equal probability, the expected weight of the returned edges is at least $\beta(d, k)w(\widehat{M} \cap \Delta(t))$, which was to be shown in the weighted case when $|T| = 1$.

The general weighted case, when the size of T is arbitrary, can be handled by a computation similar to the end of the proof of Theorem 4.9. From this, we get that $\mathbb{E}(n, d, k) \geq \beta(d, k)w(M^*)$, which completes the proof. □

It is easy to see that (14) is monotone decreasing and tends to $\frac{1}{e}$ as k and d go to infinity. This immediately implies the following.

Theorem 4.12. *Algorithm 2 outputs a permutation $s_1, \ldots, s_n$ of S and a feasible d-distance b-matching whose expected weight is at least $\frac{1}{e}$ times the weight of the heaviest d-distance matching under all permutations. This lower bound is tight.*

By Theorems 4.9 and 4.11, the expected approximation guarantees achieved by Algorithms 1 and 2 are better than e when any of d, $\frac{n}{d}$ or the largest degree in T is small. For example, if $d = 2$, then both algorithms return a 2-approximate solution in expectation.

Both algorithms are easy to de-randomize using conditional probabilities as follows. Observe that the conditional probability of an edge being added to the solution can be easily computed under the condition that the positions of some of the nodes are already fixed. Therefore, one can try to put each node to the first place, and choose the one that gives the highest (conditional) expectation. Then try each of the remaining nodes at the second position and put the best one there, and so on. The weights of the outputs of the de-randomized algorithms are clearly at least as large as the expected weight of the solutions found by the randomized algorithms.

In the rest of this section, an improved approximation algorithm for the non-cyclic case is presented. Algorithm 2 includes an edge st in the solution set if and only if t has no neighbors among the $(d-1)$ nodes in front of s. A more efficient approach is to run a greedy algorithm enumerating the edges incident to t from left to right for each $t \in T$, in other words, the modified algorithm generates a random permutation of S and executes algorithm T-Greedy as described in [1]. Clearly, the solution returned by the modified algorithm is at least as good as the one found by Algorithm 2, provided that they choose the same random permutation. We propose the following conjecture:

Conjecture 4.13. *Generating a random permutation of S, algorithm T-Greedy finds a feasible d-distance b-matching whose expected weight is at least $\frac{1}{2}\frac{d^2+d+2}{d^2+d} > \frac{1}{2}$ times the weight of the heaviest d-distance b-matching under all permutations.*

Note that the conjecture is based on an extensive computational study. We computer-verified the statement in all cases when $|S| \leq 1000$ and $d \leq 100$. Enumerating all such instances directly is hopeless, but one can design a non-trivial dynamic programming algorithm for computing the exact expected value in the case $|S| = (k-1)d+1$ and $|T| = 1$. Similarly to the proof of Theorem 4.11, this confirms the conjecture for all problem instances with $|S| \leq 1000$ and $d \leq 100$.

Note that it is not difficult to construct an example for each d in which the expected weight of the returned edges is exactly $\frac{1}{2}\frac{d^2+d+2}{d^2+d}$ times the optimum, so one cannot hope to improve the bound above.

5 Open questions

An FPT algorithm parameterized by d was given for the d-distance matching problem [1]. A straightforward generalization of this approach gives an FPT algorithm parameterized by both d and $\max_{v \in V} b(v)$ for the d-distance b-matching problem. It is not clear whether an FPT algorithm parameterized only by d exists for this problem. The d-distance matching problem was shown to be solvable in polynomial time when the size of T is a constant [1]. Is the problem polynomial-time solvable when the size of T is taken as a parameter?

It remains open whether an optimal permutation can be found when both d and the coordinates of b are constants. Improving the e-approximation algorithms for finding the best permutation — and in particular proving Conjecture 4.13 — seems to be a challenging problem.

By Theorems 4.2 and 4.3, finding a permutation maximizing the weight of the heaviest (cyclic) d-distance b-matching can be solved when $b(s) = 1$ for all s, and it is NP-hard when $b(s) = 2$ for all $s \in S$. What can we say about the cases between these two extremes? It is easy to see that the problem is solvable when $b(s) \in \{1, 2\}$ and there are only a constant number of nodes for which b is 2. A natural question is whether an FPT algorithm parameterized by the number of nodes for which b is 2 exists.

The construction given in Theorem 4.6 seems to be easy to modify to show that the optimization version of the cyclic problem is APX-hard. We do not know whether the non-cyclic optimization problem behaves differently in this regard.

The integrality gap of (LP1') is at most $(2 - \frac{1}{d})$ in the cyclic case when the size of S is divisible by $(2d - 1)$. We believe that this bound holds regardless of the size of S, but the proof of Theorem 2.1 does not seem to generalize to this case.

In the non-cyclic case, the integrality gap is at most $(2 - \frac{2}{d})$, but this does not seem to be tight. The exact value of the integrality gap remains unknown.

In a natural generalization of the problem, a bound $g(t, I) \in \mathbb{Z}_+$ is given on the number of edges induced by I and t for each $t \in T$ and for each interval I of length d in S. When $g \equiv 1$, we get back the d-distance b-matching problem. Some of the results presented in this paper also apply to other special cases of this more general setting. For example, (LP1) easily extends to the more general problem by changing the right hand-side of (1c) to $g(s_i, R_d(s_i))$, and adding $x \leq \mathbb{1}$. It is not hard to see that the bounds on the integrality gap given by Theorems 2.1 and 2.2 hold for any uniform g.

The problem has several other natural generalizations. For example, pose distance constraints on both node classes, or drop some of the distance constraints, etc., which are subjects for further research.

Motivated by the position-based scheduling problem on a single machine [18], we introduce the *position-based optimal permutation problem*, in which placing $s \in S$ at each position has an associated cost, and the goal is to find a minimum-cost permutation of S under which a perfect d-distance matching exists. When the cost function is uniform, the problem can be solved by Theorem 4.2. This approach does not seem to work for other cost functions, so it is an exciting open question whether this problem is polynomial-time solvable.

6 Acknowledgement

The author is grateful to Sára Hanna Tóth for discussions and for finding the gadget used in the proof of Theorem 3.1. This research has been implemented with the support provided by the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund, financed under the ELTE TKP 2021-NKTA-62 funding scheme, by the Ministry of Innovation and Technology NRDI Office within the framework of the Artificial Intelligence National Laboratory Program, and by the Lendület Programme of the Hungarian Academy of Sciences – grant number LP2021-1/2021.

References

[1] P. Madarasi. Matchings under distance constraints I. *Annals of Operations Research*, 305(1):137–161, 2021.

[2] M. Makai. On maximum cost $K_{t,t}$-free t-matchings of bipartite graphs. *SIAM J. Discret. Math.*, 21(2):349–360, April 2007.

[3] A. Itai, M. Rodeh, and S. Tanimoto. Some matching problems for bipartite graphs. *J. ACM*, 25:517–525, October 1978.

[4] J. Baste, D. Rautenbach, and I. Sau. Approximating maximum uniquely restricted matchings in bipartite graphs. *Discrete Applied Mathematics*, 267:30–40, 2019.

[5] K. Bérczi and L. A. Végh. Restricted b-matchings in degree-bounded graphs. In Friedrich Eisenbrand and F. Bruce Shepherd, editors, *Integer Programming and Combinatorial Optimization*, pages 43–56, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[6] M. Fürst and D. Rautenbach. On some hard and some tractable cases of the maximum acyclic matching problem. *Annals of Operations Research*, 279(1-2):291–300, 2019.

[7] Gy. Pap. Alternating paths revisited II: restricted b-matchings in bipartite graphs. *EGRES Technical Report TR-2005-13*, 2005.

[8] T. Zeitlhofer and B. Wess. List-coloring of interval graphs with application to register assignment for heterogeneous register-set architectures. *Signal Processing*, 83:1411–1425, 2003.

[9] K. I. Aardal, S. P. M. van Hoesel, A. M. C. A. Koster, C. Mannino, and A. Sassano. Models and solution techniques for frequency assignment problems. *Annals of Operations Research*, 153(1):79–129, September 2007.

[10] A. Frank. *Connections in Combinatorial Optimization*. Oxford University Press, 2011.

[11] É. Tardos. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research*, 34(2):250–256, 1986.

[12] P. Madarasi. The simultaneous assignment problem, 2021.

[13] V. Kann. Maximum bounded 3-dimensional matching is MAX SNP-complete. *Information Processing Letters*, 37(1):27–35, 1991.

[14] M. Chlebík and J. Chlebíková. Complexity of approximating bounded variants of optimization problems. *Theoretical Computer Science*, 354(3):320 – 338, 2006. Foundations of Computation Theory (FCT 2003).

[15] P. Madarasi. The distance matching problem. In Mourad Baïou, Bernard Gendron, Oktay Günlük, and A. Ridha Mahjoub, editors, *Combinatorial Optimization*, pages 202–213, Cham, 2020. Springer International Publishing.

[16] K. Bérczi and T. Schwarcz. Complexity of packing common bases in matroids. *Mathematical Programming*, 188(1):1–18, 2021.

[17] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[18] M. Horváth and T. Kis. Polyhedral results for position-based scheduling of chains on a single machine. *Annals of Operations Research*, 284(1):283–322, 2020.